AN1078

Sensorless Field Oriented Control of a PMSM

Authors: Jorge Zambada and Debraj Deb
Microchip Technology Inc.

INTRODUCTION

Designers can expect environmental demands to continue to drive the need for advanced motor control techniques that produce energy efficient air conditioners, washing machines and other home appliances. Until now, sophisticated motor control solutions have only been available from proprietary sources. However, the implementation of advanced, cost-effective motor control algorithms is now a reality, thanks to the new generation of Digital Signal Controllers (DSCs).

An air conditioner, for example, requires fast response for speed changes in the motor. Advanced motor control algorithms are needed to produce quieter units that are more energy efficient. Field Oriented Control (FOC) has emerged as the leading method to achieve these environmental demands.

This application note discusses the implementation of a sensorless FOC algorithm for a Permanent Magnet Synchronous Motor (PMSM) using the Microchip dsPIC® DSC family.

Why Use the FOC Algorithm?

The traditional control method for BLDC motors drives the stator in a six-step process, which generates oscillations on the produced torque. In six-step control, a pair of windings is energized until the rotor reaches the next position, and then the motor is commutated to the next step. Hall sensors determine the rotor position to electronically commutate the motor. Advanced sensorless algorithms use the back-EMF generated in the stator winding to determine the rotor position.

The dynamic response of six-step control (also called trapezoidal control) is not suitable for washing machines because the load is changing dynamically within a wash cycle, and varies with different loads and the selected wash cycle. Further, in a front load washing machine, the gravitational power works against the motor load when the load is on the top side of the drum. Only advanced algorithms such as FOC can handle these dynamic load changes.

This application note focuses on the PMSM-based sensorless FOC control of appliances because this control technique offers the greatest cost benefit in appliance motor control. The sensorless FOC technique also overcomes restrictions placed on some applications that cannot deploy position or speed sensors because the motor is flooded, or because of wire harness placement constraints. With a constant rotor magnetic field produced by a permanent magnet on the rotor, the PMSM is very efficient when used in an appliance. In addition, its stator magnetic field is generated by sinusoidal distribution of windings. When compared to induction motors, a PMSM is powerful for its size. It is also electrically less noisy than a DC motor, since brushes are not used.

Why Use Digital Signal Controllers for Motor Control?

dsPIC DSCs are suitable for appliances like washing machines and air conditioner compressors because they incorporate peripherals that are ideally suited for motor control, such as:

- Pulse-Width Modulation (PWM)
- Analog-to-Digital Converter (ADC)
- Quadrature Encoder Interface (QEI)

When performing controller routines and implementing digital filters, dsPIC DSCs enable designers to optimize code because MAC instructions and fractional operations can be executed in a single cycle. Also, for operations that require saturation capabilities, the dsPIC DSCs help avoid overflows by offering hardware saturation protection.

The dsPIC DSCs need fast and flexible Analog-to-Digital (A/D) conversion for current sensing—a crucial function in motor control. The dsPIC DSCs feature ADCs that can convert input samples at 1 Msps rates, and handle up to four inputs simultaneously. Multiple trigger options on the ADCs enable use of inexpensive current sense resistors to measure winding currents. For example, the ability to trigger A/D conversions with the PWM module allows inexpensive current sensing circuitry to sense inputs at specific times (switching transistors allow current to flow through sense resistors).

MOTOR CONTROL WITH DIGITAL SIGNAL CONTROLLERS

The dsPIC DSC Motor Control family is specifically designed to control the most popular types of motors, including:

- AC Induction Motor (ACIM)
- Brushed DC Motor (BDC)
- Brushless DC Motor (BLDC)
- Permanent Magnet Synchronous Motor (PMSM)

Several application notes have been published based on the dsPIC DSC motor control family (see the **"References"** section). These application notes are available on the Microchip web site (www.microchip.com).

This application note demonstrates how the dsPIC DSC takes advantage of peripherals specifically suited for motor control (motor control PWM and high-speed ADC) to execute sensorless field oriented control of a PMSM. The DSP engine of the dsPIC DSC supports the necessary fast mathematical operations.

Data Monitoring and Control Interface

The Data Monitor and Control Interface (DMCI) provides quick dynamic integration with MPLAB® IDE for projects in which operational constraints of the application depend on variable control of range values, on/off states or discrete values. If needed, application feedback can be represented graphically. Examples include motor control and audio processing applications.

The DMCI provides:

- Nine slider controls and nine boolean (on/off) controls (see Figure 1)
- 35 input controls (see Figure 2)
- Four graphs (see Figure 3)

The interface provides project-aware navigation of program symbols (variables) that can be dynamically assigned to any combination of slider, direct input or boolean controls. The controls can then be used interactively to change values of program variables within MPLAB IDE. The graphs can be dynamically configured for viewing program generated data.

Note:	The characteristics of the DMCI tool are subject to change. This description of the DMCI tool is accurate at the date of publication.

Application Highlights

The purpose of this application note is to illustrate a software-based implementation of sensorless, field oriented control for PMSM using Microchip digital signal controllers.

The control software offers these features:

- Implements vector control of a PMSM.
- Position and speed estimation algorithm. eliminates the need for position sensors.
- Speed range tested from 500 to 17000 RPM.
- With a 50 µs control loop period, the software requires approximately 21 MIPS of CPU overhead (about 2/3 of the total available CPU).
- The application requires 450 bytes of data memory storage. With the user interface, approximately 6 Kbytes of program memory are required. The memory requirements of the application allow it to run on the dsPIC33FJ12MC202, which is the smallest and most cost-effective dsPIC33F device at the time of this writing.
- An optional diagnostics mode can be enabled to allow real-time observation of internal program variables on an oscilloscope. This feature facilitates control loop adjustment.

FIGURE 1: DYNAMIC DATA CONTROL INTERFACE

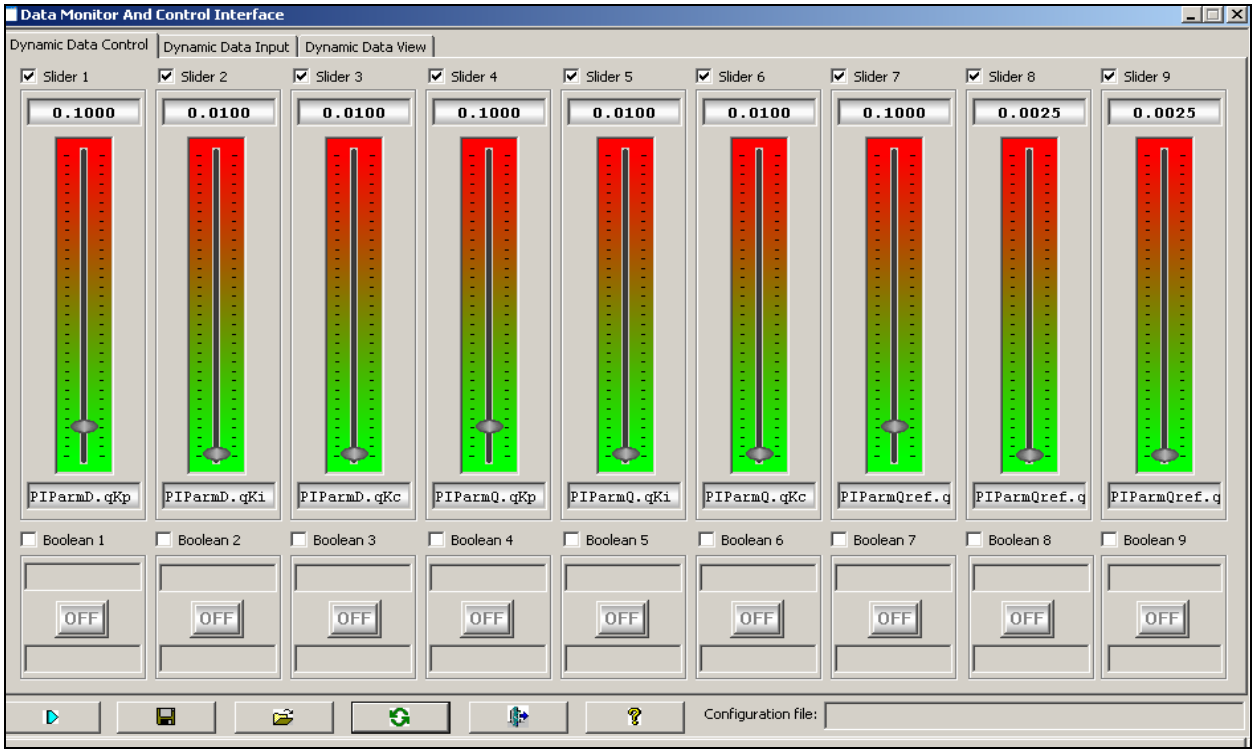


FIGURE 2: USER-DEFINED DATA INPUT CONTROLS

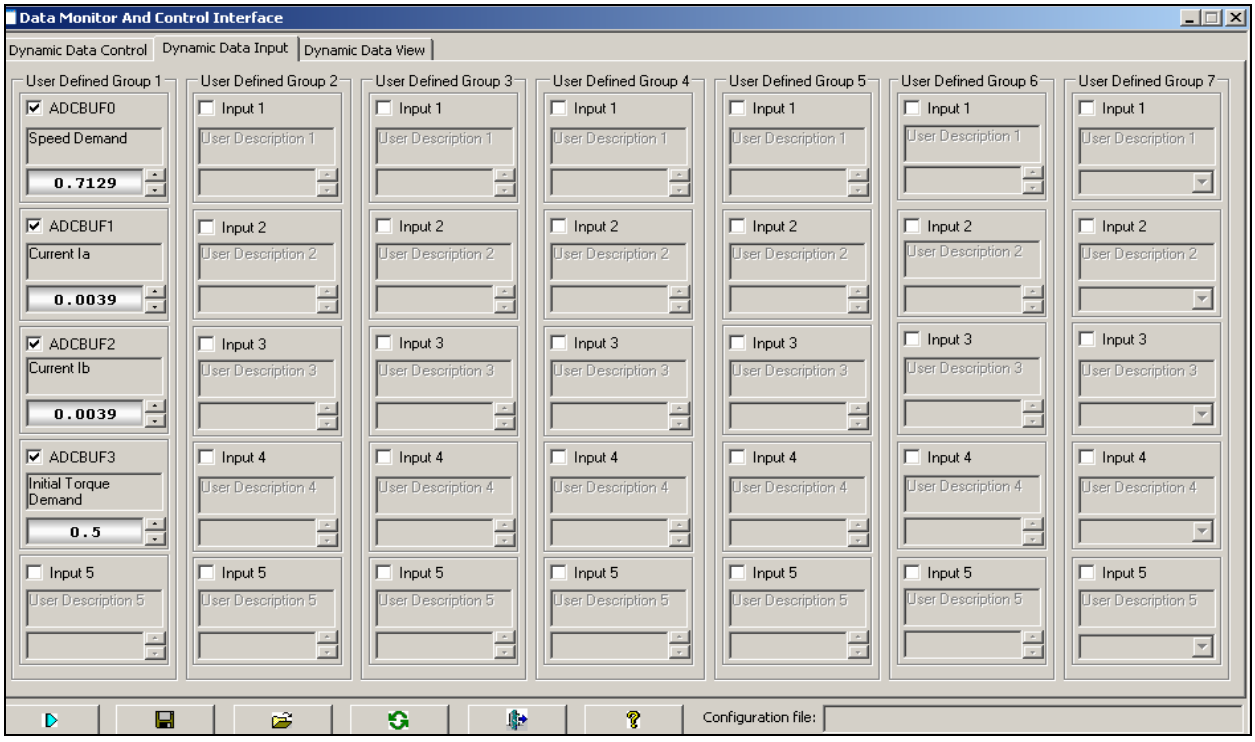
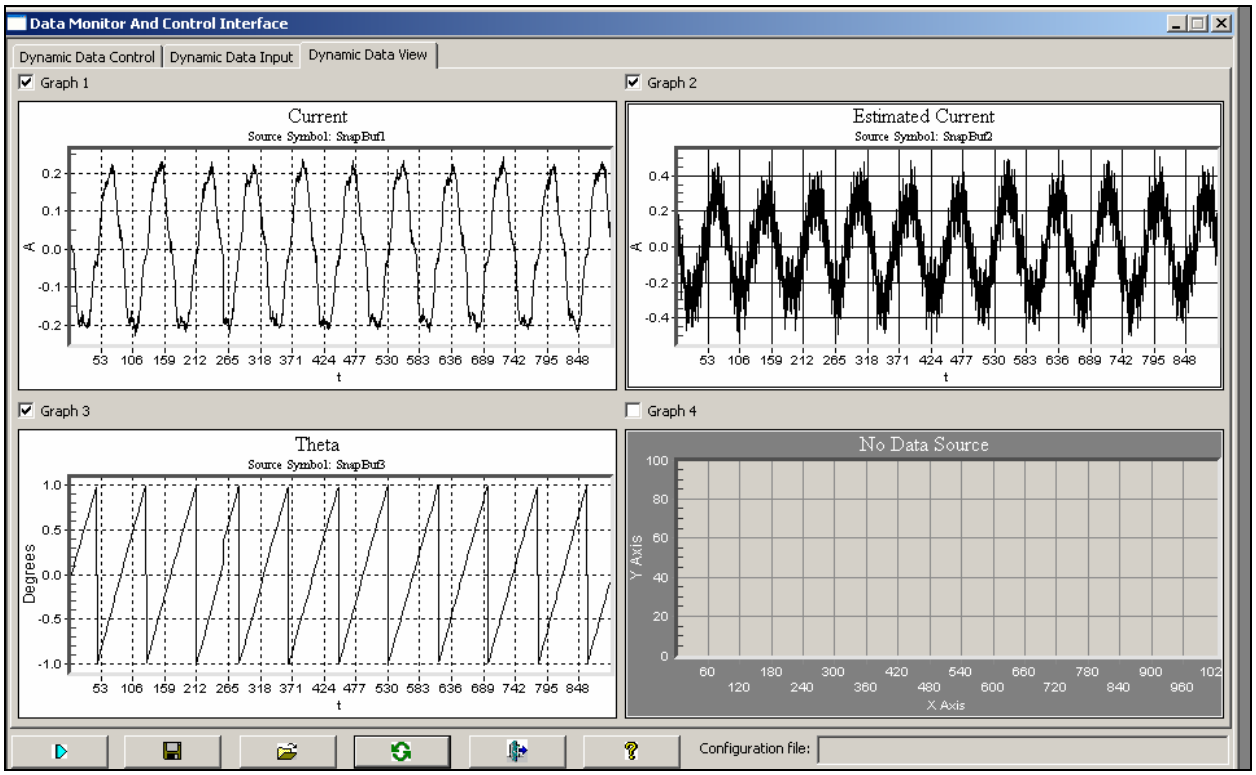


FIGURE 3: GRAPHICAL DATA VIEW



SYSTEM OVERVIEW

As shown in Figure 4, there are no position sensors attached to the motor shaft. Instead, low-inductance shunt resistors, which are part of the inverter are used for current measurements on the motor. A 3-phase inverter is used as the power stage to drive motor windings. Current sensing and fault generation circuitry built into the power inverter protects the overall system against over currents.

Figure 5 illustrates how the 3-phase topology, as well as the current detection and fault generation circuitry, are implemented.

The first transistor shown on the left side of the inverter is used for Power Factor Correction (PFC), which is not part of this application note.

The hardware that is referred to in this application note are the dsPICDEM™ MCLV Development Board (DM330021) (up to 50 VDC) and the dsPICDEM™ MCHV Development Board (DM330023) (up to 400 VDC), which are available from the Microchip web site (www.microchip.com).

FIGURE 4: SYSTEM OVERVIEW

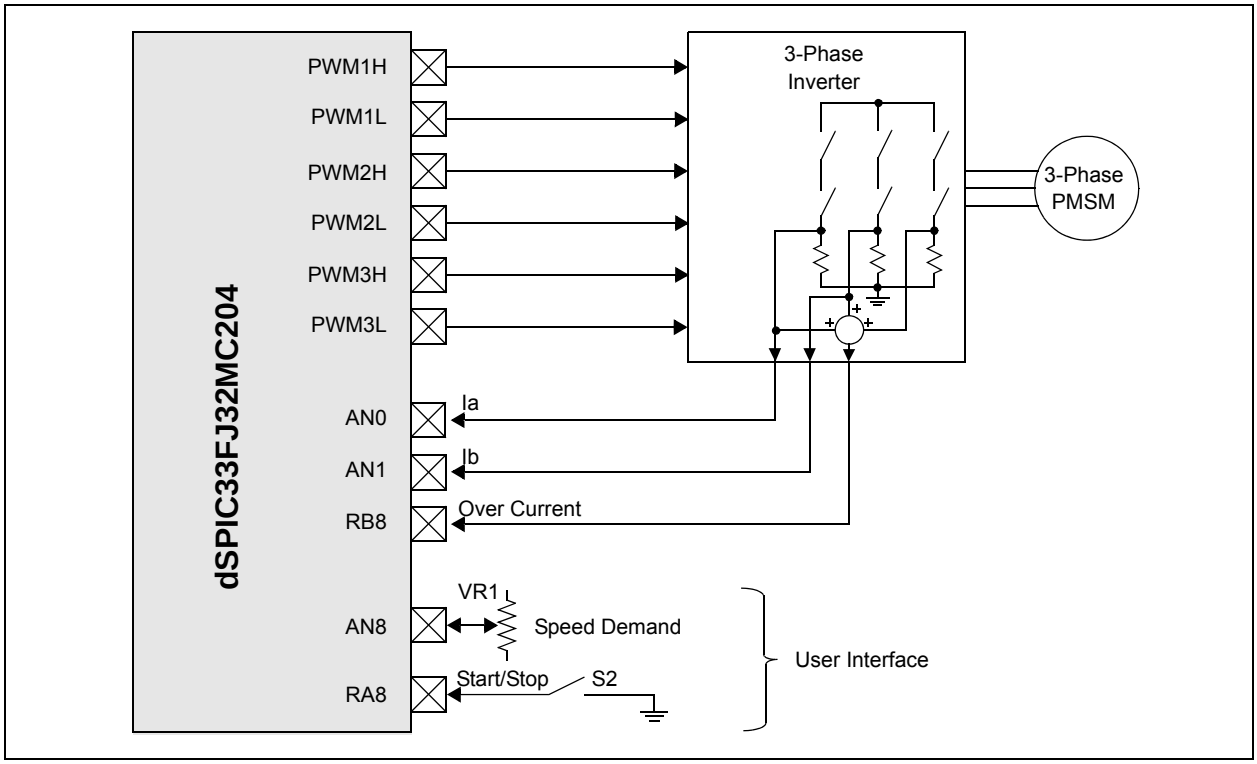
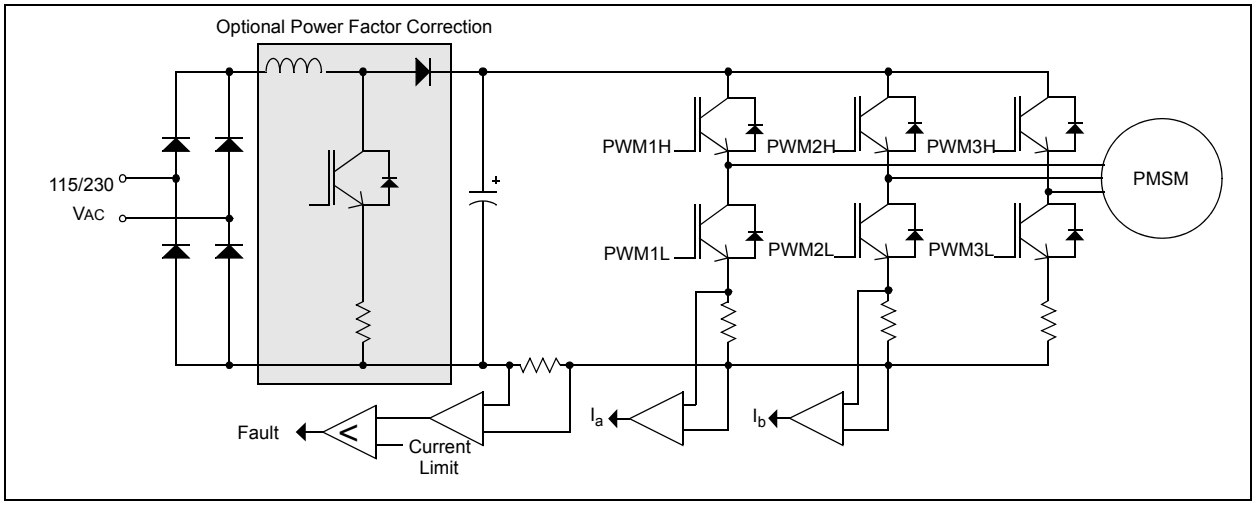


FIGURE 5: 3-PHASE TOPOLOGY



FIELD ORIENTED CONTROL

A Matter of Perspective

One way to understand how FOC (sometimes referred to as vector control) works is to form a mental image of the coordinate reference transformation process. If you picture an AC motor operation from the perspective of the stator, you see a sinusoidal input current applied to the stator. This time variant signal generates a rotating magnetic flux. The speed of the rotor is a function of the rotating flux vector. From a stationary perspective, the stator currents and the rotating flux vector look like AC quantities.

Now, imagine being inside the motor and running alongside the spinning rotor at the same speed as the rotating flux vector generated by the stator currents. If you were to look at the motor from this perspective during steady state conditions, the stator currents look like constant values, and the rotating flux vector is stationary.

Ultimately, you want to control the stator currents to obtain the desired rotor currents (which cannot be measured directly). With coordinate reference transformation, the stator currents can be controlled like DC values using standard control loops.

Vector Control Summary

The indirect vector control process can be summarized as follows:

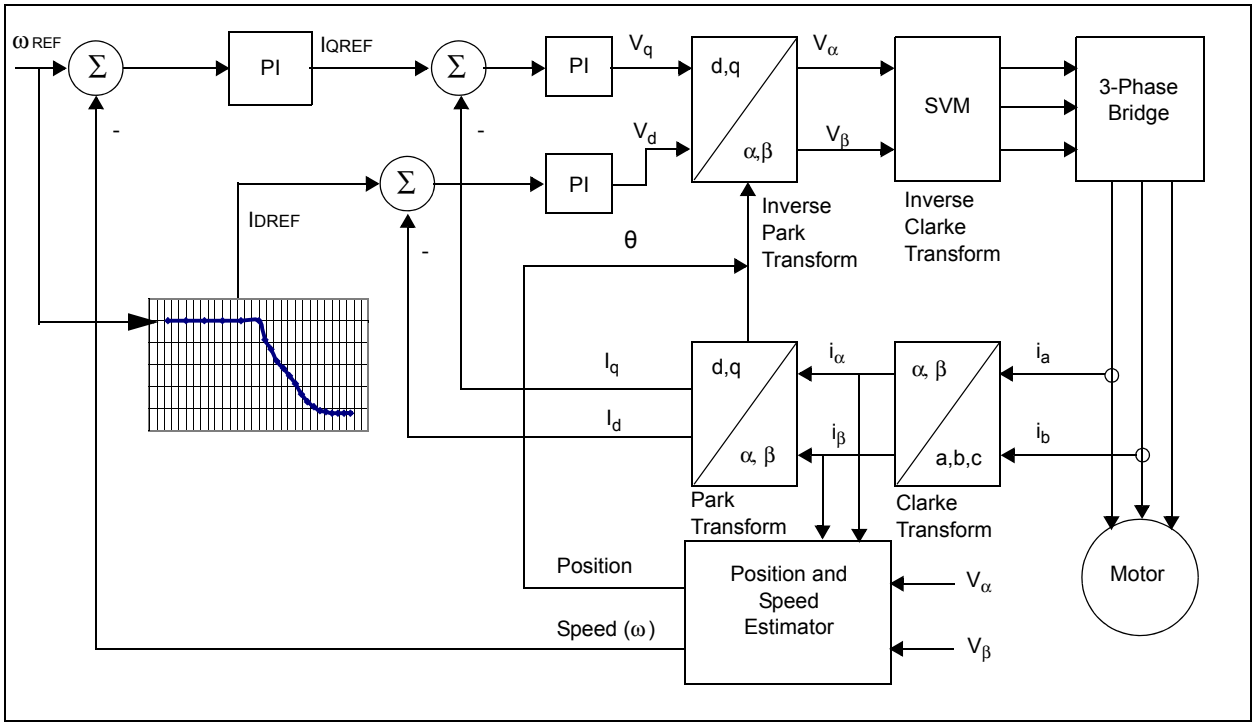
1. The 3-phase stator currents are measured. These measurements provide values i_a and i_b. I_c is calculated by the following equation:

   $i_a + i_b + i_c = 0$.

2. The 3-phase currents are converted to a two-axis system. This conversion provides the variables i_α and i_β from the measured i_a and i_b and the calculated i_c values. i_α and i_β are time-varying quadrature current values as viewed from the perspective of the stator.
3. The two-axis coordinate system is rotated to align with the rotor flux using a transformation angle calculated at the last iteration of the control loop. This conversion provides the I_d and I_q variables from i_α and i_β. I_d and I_q are the quadrature currents transformed to the rotating coordinate system. For steady state conditions, I_d and I_q are constant.
4. Error signals are formed using I_d, I_q and reference values for each.
   - The I_d reference, controls rotor magnetizing flux
   - The I_q reference, controls the torque output of the motor
   - The error signals are input to PI controllers
   - The output of the controllers provide V_d and V_q, which are voltage vector that will be sent to the motor
5. A new transformation angle is estimated where v_α, v_β, i_α and i_β are the inputs. The new angle guides the FOC algorithm as to where to place the next voltage vector.
6. The V_d and V_q output values from the PI controllers are rotated back to the stationary reference frame using the new angle. This calculation provides the next quadrature voltage values v_α and v_β.
7. The v_α and v_β values are transformed back to 3-phase values v_a, v_b and v_c. The 3-phase voltage values are used to calculate new PWM duty cycle values that generate the desired voltage vector. The entire process of transforming, PI iteration, transforming back and generating PWM is illustrated in Figure 6.

The next sections of this application note describe these steps in greater detail.

FIGURE 6: VECTOR CONTROL BLOCK DIAGRAM



COORDINATE TRANSFORMS

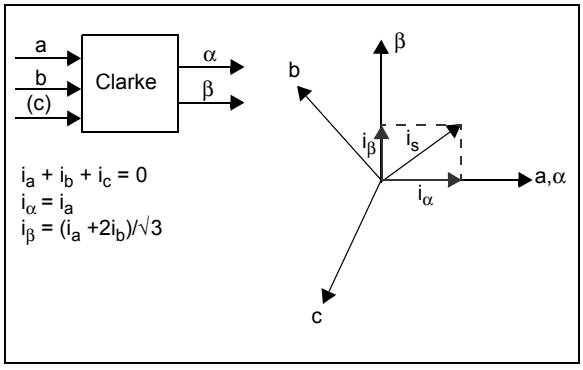
Through a series of coordinate transforms, you can indirectly determine and control the time invariant values of torque and flux with classic PI control loops. The process begins by measuring the 3-phase motor currents. In practice, the instantaneous sum of the three current values is zero. Therefore, by measuring only two of the three currents, you can determine the third. Because of this fact, hardware cost can be reduced by the expense of the third current sensor.

A single shunt implementation for 3-phase current measurement is also possible with the dsPIC DSC. Refer to the *AN1299, "Single-Shunt Three-Phase Current Reconstruction Algorithm for Sensorless FOC of a PMSM"* (DS01299) for detailed description of single shunt algorithm.

Clarke Transform

The first coordinate transform, called the Clarke Transform, moves a three-axis, two-dimensional coordinate system, referenced to the stator, onto a two-axis system, keeping the same reference (see Figure 7, where i_a, i_b and i_c are the individual phase currents).

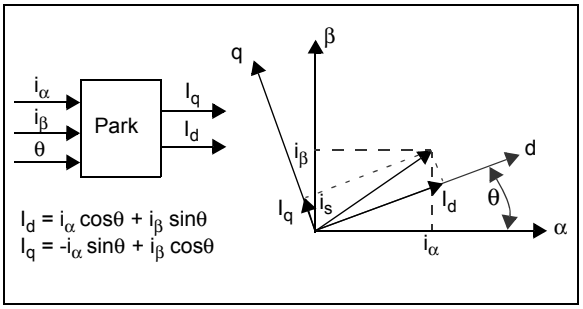
FIGURE 7: CLARKE TRANSFORM



Park Transform

At this point, you have the stator current represented on a two-axis orthogonal system with the axis called α-β. The next step is to transform into another two-axis system that is rotating with the rotor flux. This transformation uses the Park Transform, as illustrated in Figure 8. This two-axis rotating coordinate system is called the d-q axis. θ represents the rotor angle.

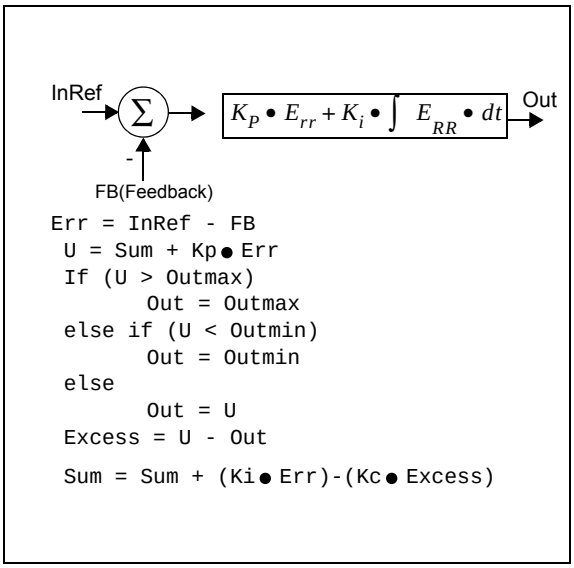
FIGURE 8: PARK TRANSFORM



PI Control

Three PI loops are used to control three interactive variables independently. The rotor speed, rotor flux and rotor torque are each controlled by a separate PI module. The implementation is conventional and includes term (Kc • Excess) to limit integral windup, as illustrated in Figure . Excess is calculated by subtracting the unlimited output (U) and limited output (Out). The term Kc multiplies the Excess and limits the accumulated integral portion (Sum).

FIGURE 9: PI CONTROL



PID CONTROLLER BACKGROUND

A complete discussion of Proportional Integral Derivative (PID) controllers is beyond the scope of this application note; however, this section provides you with some basics of PID operation.

A PID controller responds to an error signal in a closed control loop and attempts to adjust the controlled quantity to achieve the desired system response. The controlled parameter can be any measurable system quantity such as speed, torque or flux. The benefit of the PID controller is that, it can be adjusted empirically by varying one or more gain values and observing the change in the system response.

A digital PID controller is executed at a periodic sampling interval. It is assumed that the controller is executed frequently enough so that the system can be properly controlled. The error signal is formed by subtracting the desired setting of the parameter to be controlled from the actual measured value of that parameter. The sign of the error indicates the direction of change required by the control input.

The Proportional (P) term of the controller is formed by multiplying the error signal by a P gain, causing the PID controller to produce a control response that is a function of the error magnitude. As the error signal becomes larger, the P term of the controller becomes larger to provide more correction.

The effect of the P term tends to reduce the overall error as time elapses. However, the effect of the P term diminishes as the error approaches zero. In most systems, the error of the controlled parameter gets very close to zero but does not converge. The result is a small remaining steady state error.

The Integral (I) term of the controller is used to eliminate small steady state errors. The I term calculates a continuous running total of the error signal. Therefore, a small steady state error accumulates into a large error value over time. This accumulated error signal is multiplied by an I gain factor and becomes the I output term of the PID controller.

The Differential (D) term of the PID controller is used to enhance the speed of the controller and responds to the rate of change of the error signal. The D term input is calculated by subtracting the present error value from a prior value. This delta error value is multiplied by a D gain factor that becomes the D output term of the PID controller.

The D term of the controller produces more control output as the system error changes more rapidly. Not all PID controllers will implement the D or, less commonly, the I terms. For example, this application does not use the D terms due to the relatively slow response time of motor speed changes. In this case, the D term could cause excessive changes in PWM duty cycle that could affect the operation of the algorithms and produce over current trips.

Adjusting the PID Gains

The P gain of a PID controller sets the overall system response. When you first tune a controller, set the I and D gains to zero. You can then increase the P gain until the system responds well to set point changes without excessive overshoot or oscillations. Using lower values of P gain will 'loosely' control the system, while higher values will give 'tighter' control. At this point, the system will probably not converge to the set point.

After you select a reasonable P gain, you can slowly increase the I gain to force the system error to zero. Only a small amount of I gain is required in most systems. The effect of the I gain, if large enough, can overcome the action of the P term, slow the overall control response and cause the system to oscillate around the set point. If oscillation occurs, reducing the I gain and increasing the P gain will usually solve the problem.

This application includes a term to limit integral windup, which occurs if the integrated error saturates the output parameter. Any further increase in the integrated error does not affect the output. The accumulated error, when it does decrease, will have to fall (or unwind) to below the value that caused the output to saturate. The Kc coefficient limits this unwanted accumulation. For most situations, this coefficient can be set equal to Ki.

All three controllers have a maximum value for the output parameter. These values can be found in the `UserParms.h` file and are set by default to avoid saturation in the `SVGen()` routine.

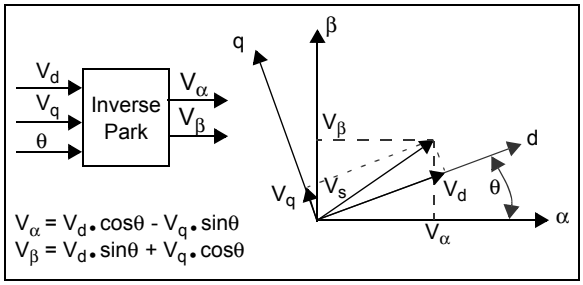
Control Loop Dependencies

There are three interdependent PI control loops in this application. The outer loop controls the motor velocity. The two inner loops control the transformed motor currents, I_d and I_q. As mentioned previously, the I_d loop is responsible for controlling flux, and the I_q value is responsible for controlling the motor torque.

Inverse Park

After the PI iteration, you have two voltage component vectors in the rotating d-q axis. You will need to go through complementary inverse transforms to get back to the 3-phase motor voltage. First, you transform from the two-axis rotating d-q frame to the two-axis stationary frame α-β. This transformation uses the Inverse Park Transform, as illustrated in Figure 10.

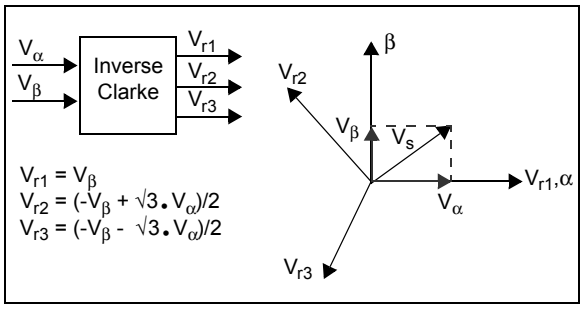
FIGURE 10: INVERSE PARK



Inverse Clarke

The next step is to transform from the stationary two-axis α-β frame to the stationary three-axis, 3-phase reference frame of the stator. Mathematically, this transformation is accomplished with the Inverse Clark Transform, as illustrated in Figure 11.

FIGURE 11: INVERSE CLARKE



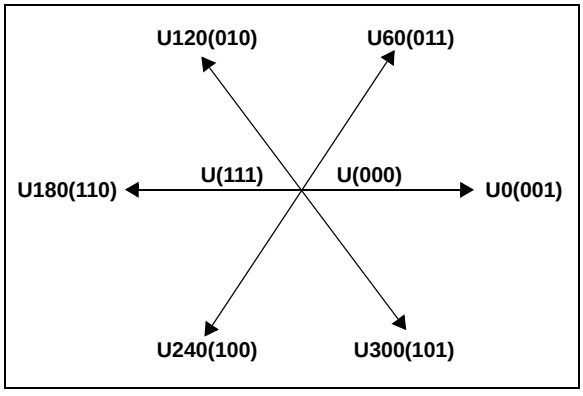
Space Vector Modulation (SVM)

The final step in the vector control process is to generate pulse-width modulation signals for the 3-phase motor voltage signals. If you use Space Vector Modulation (SVM) techniques, the process of generating the pulse width for each of the three phases is reduced to a few simple equations. In this implementation, the Inverse Clarke Transform has been folded into the SVM routine, which further simplifies the calculations.

Each of the three inverter outputs can be in one of two states. The inverter output can be connected to either the plus (+) bus rail or the minus (-) bus rail, which allows for 2^3 = 8 possible states of the output as shown in Table 1.

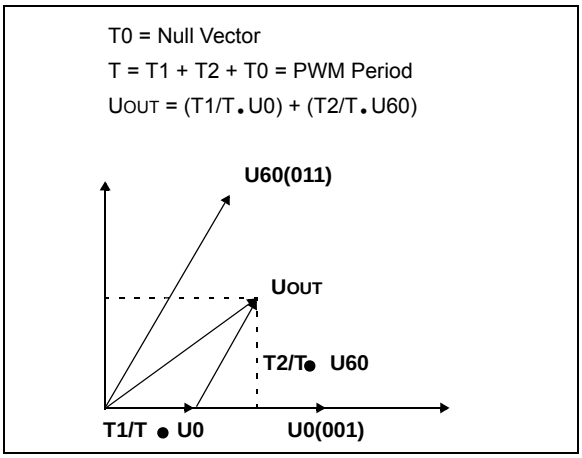
The two states in which all three outputs are connected to either the plus (+) bus or the minus (-) bus are considered null states because there is no line-to-line voltage across any of the phases. These are plotted at the origin of the SVM star. The remaining six states are represented as vectors with 60 degree rotation between each state, as shown in Figure 12.

FIGURE 12: SVM



The process of SVM allows the representation of any resultant vector by the sum of the components of the two adjacent vectors. In Figure 13, UOUT is the desired resultant. It lies in the sector between U60 and U0. If during a given PWM period T, U0 is output for T1/T and U60 is output for T2/T, the average for the period will be UOUT.

FIGURE 13: AVERAGE SVM



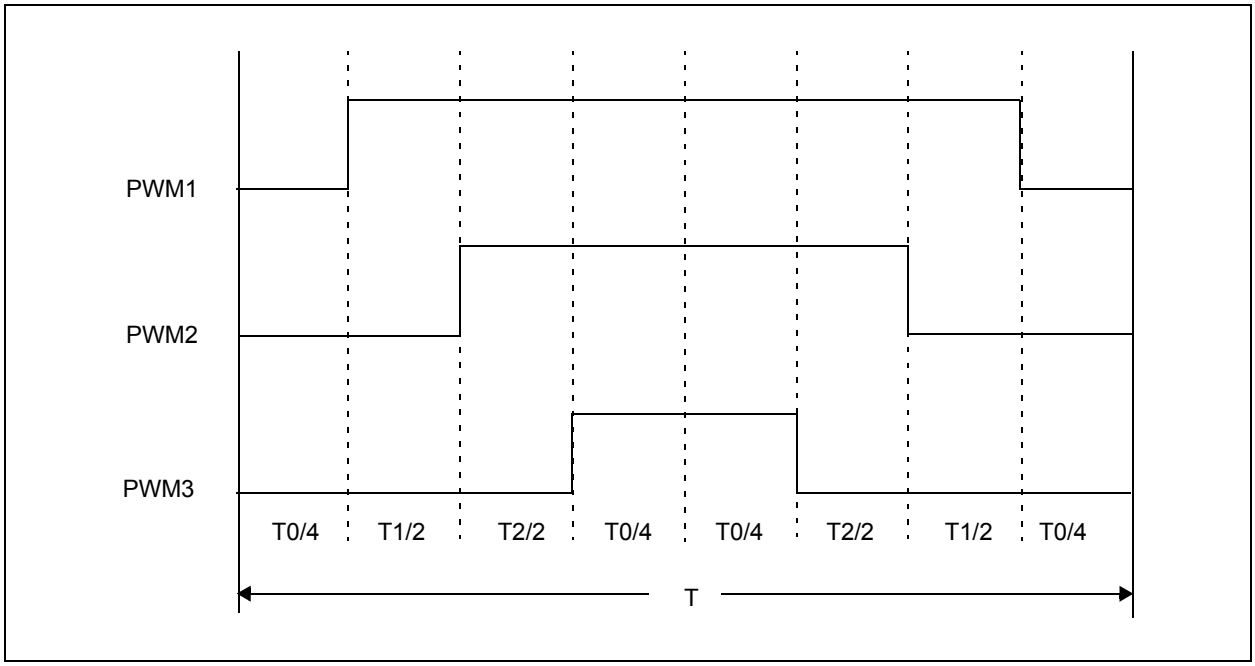
T0 represents a time where no effective voltage is applied into the windings; that is, where a null vector is applied. The values for T1 and T2 can be extracted with no extra calculations by using a modified Inverse Clark transformation. If you reverse V_α and V_β, a reference axis is generated that is shifted by 30 degrees from the SVM star. As a result, for each of the six segments, one axis is exactly opposite that segment and the other two axes symmetrically bound the segment. The values of the vector components along those two bounding axis are equal to T1 and T2. See the `CalcRef.s` and `SVGen.s` files in the source code for details of the calculations.

You can see from Figure 14 that for the PWM period T, the vector T1 is output for T1/T and the vector T2 is output for T2/T. During the remaining time the null vectors are output. The dsPIC DSC is configured for center-aligned PWM, which forces symmetry about the center of the period. This configuration produces two pulses line-to-line during each period. The effective switching frequency is doubled, reducing the ripple current while not increasing the switching losses in the power devices.

TABLE 1: SPACE VECTOR MODULATION INVERTER STATES

Phase C	Phase B	Phase A	V_{ab}	V_{bc}	V_{ca}	V_{ds}	V_{qs}	Vector
0	0	0	0	0	0	0	0	U(000)
0	0	1	V_{DC}	0	$-V_{DC}$	$2/3V_{DC}$	0	U_0
0	1	1	0	V_{DC}	$-V_{DC}$	$V_{DC}/3$	$V_{DC}/3$	U_{60}
0	1	0	$-V_{DC}$	V_{DC}	0	$-V_{DC}/3$	$V_{DC}/3$	U_{120}
1	1	0	$-V_{DC}$	0	V_{DC}	$-2V_{DC}/3$	0	U_{180}
1	0	0	0	$-V_{DC}$	V_{DC}	$-V_{DC}/3$	$-V_{DC}/3$	U_{240}
1	0	1	V_{DC}	$-V_{DC}$	0	$V_{DC}/3$	$-V_{DC}/3$	U_{300}
1	1	1	0	0	0	0	0	U(111)

FIGURE 14: PWM FOR PERIOD T



SENSORLESS FOC FOR PMSM

An important part of the algorithm is how to calculate the commutation angle needed for FOC. This section of the application note explains the process of estimating commutation angle (θ) and motor speed (ω).

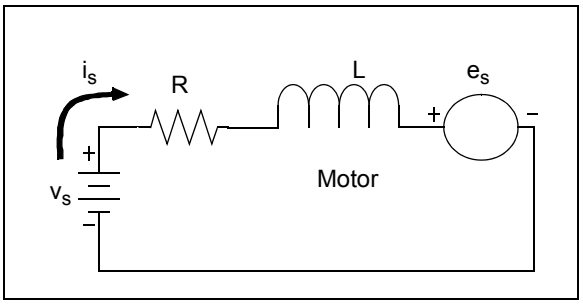
The sensorless control technique implements the FOC algorithm by estimating the position of the motor without using position sensors. Figure 16 illustrates a simplified block diagram of the position estimator function.

Motor position and speed are estimated based on measured currents and calculated voltages.

Motor Model

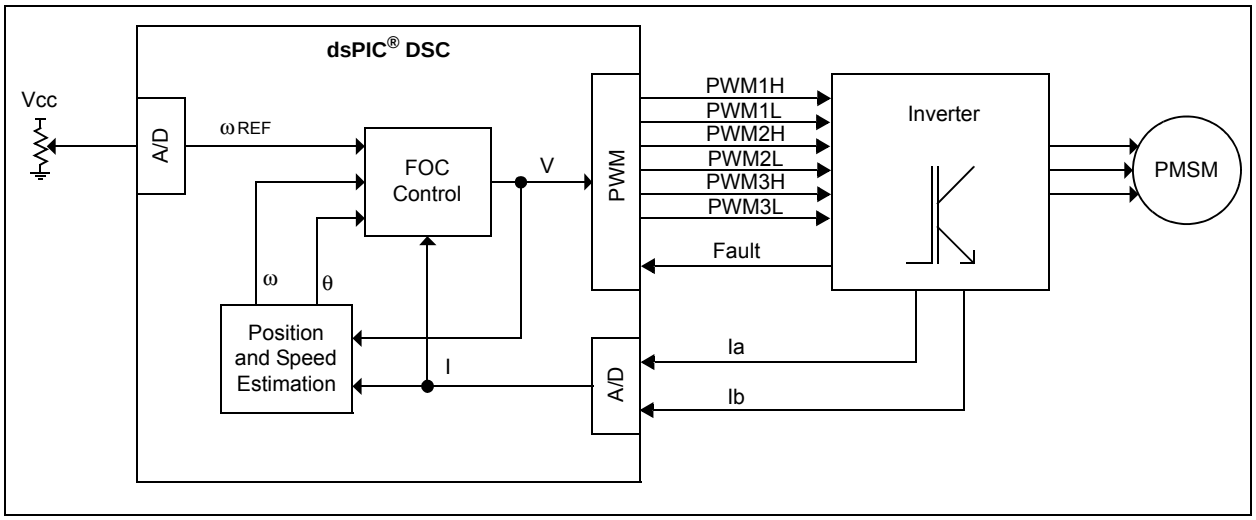
You can estimate the PMSM position by using a model of a DC Motor, which can be represented by winding resistance, winding inductance and back-EMF, as shown in Figure 15.

FIGURE 15: MOTOR MODEL



From the motor model, the input voltage can be obtained by Equation 1.

FIGURE 16: POSITION ESTIMATOR FUNCTION BLOCK DIAGRAM



EQUATION 1: DIGITIZED MOTOR MODEL

$$v_s = Ri_s + L\frac{d}{dt}i_s + e_s$$

Where:

i_s = Motor Current Vector
v_s = Input Voltage Vector
e_s = Back-EMF Vector
R = Winding Resistance
L = Winding Inductance
T_S = Control Period

Motor current is obtained by solving for i_s:

$$\frac{d}{dt}i_s = \left(-\frac{R}{L}\right)i_s + \frac{1}{L}(v_s - e_s)$$

In the digital domain, this equation becomes:

$$\frac{i_s(n+1) - i_s(n)}{T_s} = \left(-\frac{R}{L}\right)i_s(n) + \frac{1}{L}(v_s(n) - e_s(n))$$

Solving for i_s:

$$i_s(n+1) = \left(1 - Ts \bullet \frac{R}{L}\right)i_s(n) + \frac{T_s}{L}(v_s(n) - e_s(n))$$

$$F = 1 - Ts \bullet \frac{R}{L}$$

$$G = \frac{T_S}{L}$$

Calculating F and G Parameters

The motor model has two parameters that need to be modified for a particular motor. These two parameters are F and G gains, where:

EQUATION 2:

$$F = 1 - Ts \bullet \frac{R}{L}$$

$$G = \frac{T_S}{L}$$

Constants R and L are measured using a simple multimeter. For example, if a line to line resistance is measured, the R used for F and G gains is the measurement divided by two, since the phase resistance is needed. The same procedure applies to inductance calculation L. For example, the Hurst motor is run with algorithm at 20 kHz, where the line-to-line resistance measured is 5.34Ω, and line-to-line inductance measured is 3.84 mH, then the motor model parameters are:

EQUATION 3:

$$F = 1 - Ts \bullet \frac{R}{L} = \left(1 - \frac{1}{20\ KHz}\right) \bullet \frac{(5.34\Omega) / 2}{(3.84mH) / 2} = 0.9304$$

$$G = \frac{Ts}{L} = \frac{(1 / 20)kHz}{(3.84mh) / 2} = 0.026$$

Current Observer

The position and speed estimator is based on a current observer. This observer is a digitized model of the motor, as represented by Equation 1. Variables and constants include:

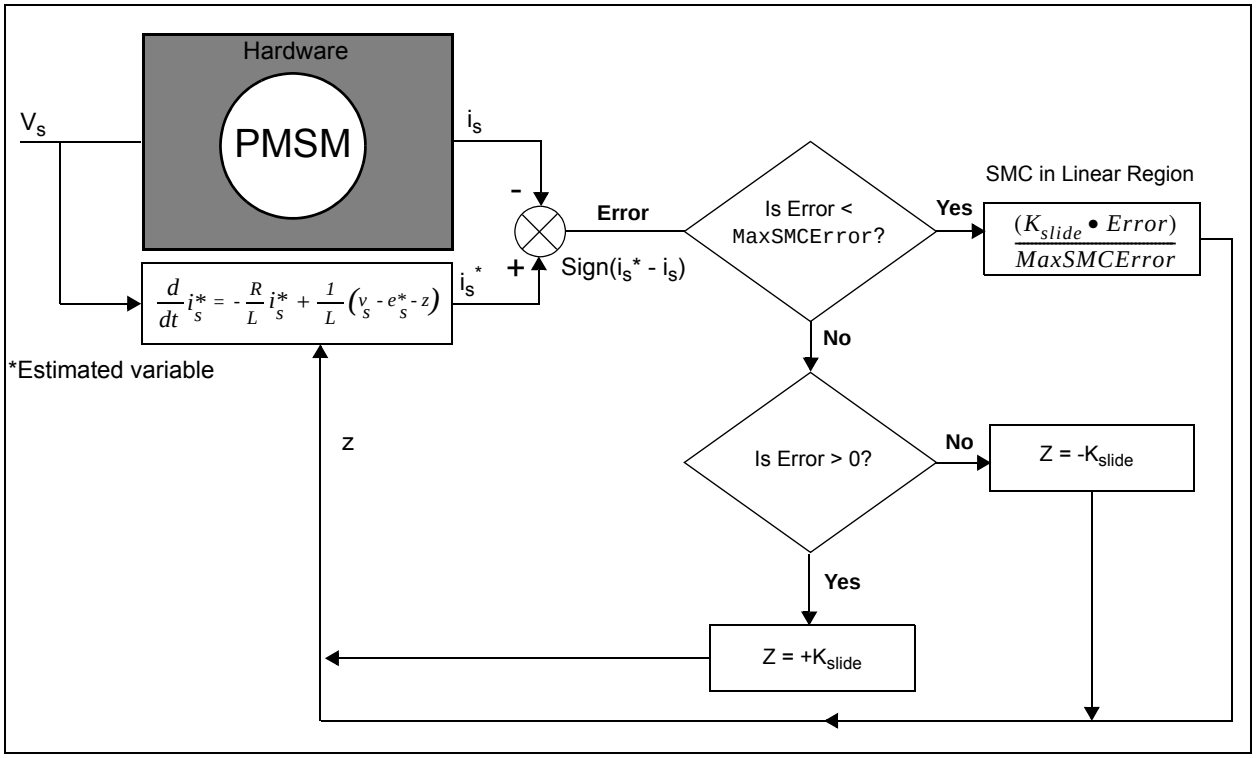
- Motor Phase Current (i_s)
- Input voltage (v_s)
- Back-EMF (e_s)
- Winding resistance (R)
- Winding inductance (L)
- Control period (T_s)
- Output Correction Factor Voltage (z)

The digitized model provides a software representation of the hardware. However, in order to match the measured current and estimated current, the digitized motor model needs to be corrected using the closed loop, as shown in Figure 17.

Considering two motor representations, one in hardware (shaded area) and one in software, with the same input (v_s) fed into both systems, and matching the measured current (i_s) with estimated current (i_s*)from the model, we can presume that back-EMF (e_s*) from our digitized model is the same as the back-EMF (e_s) from the motor.

Note: * implies the estimated variable.

FIGURE 17: CURRENT OBSERVER BLOCK DIAGRAM



The sliding-mode controller has a limit value `MaxSMCError` defined in `UserParms.h`. When the error value is lesser than the `MaxSMCError`, the output of sliding-mode controller works in the linear range as given by the equation beneath the PMSM block in Figure 17.

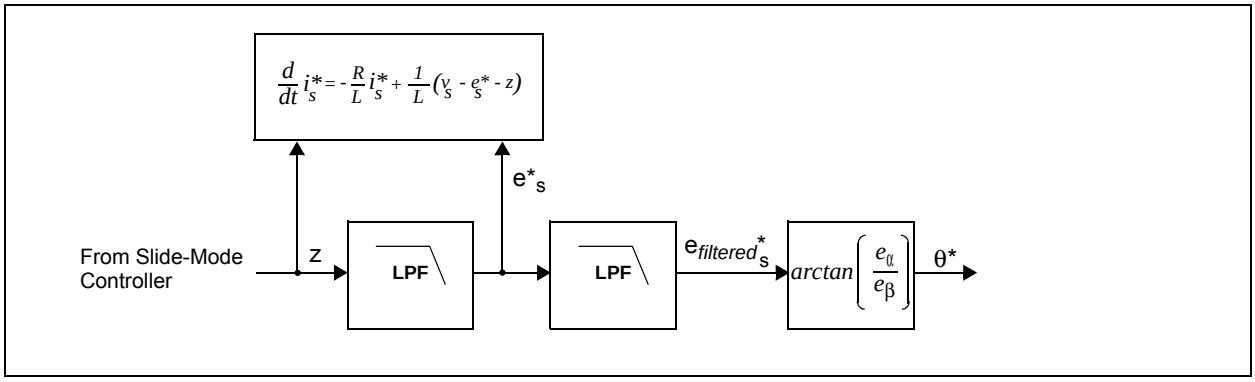
For an error value outside of the linear range, the output of the sliding mode controller is ($+K_{slide}$)/($-K_{slide}$) depending on the sign of the error.

A slide-mode controller, or SMC, is used to compensate the digitized motor model. A SMC consists of a summation point that calculates the sign of the error between measured current from the motor and estimated current from the digitized motor model. The computed sign of the error (+1 or -1) is multiplied by a SMC gain (K). The output of the SMC controller is the correction factor (Z). This gain is added to the voltage term from the digitized model, and the process repeats every control cycle until the error between measured current (i_s) and estimated current (i_s*) is zero (i.e., until the measured current and estimated current match).

Back-EMF Estimation

After compensating the digitized model, you have a motor model with the same variable values for the input voltage (V_s) and for current (i_s*). Once the digitized model is compensated, the next step is to estimate back-EMF (e_s*) by filtering the correction factor (Z), as shown in Figure 18. The back-EMF estimation (es*) is fed back to the model to update the variable e_s* after every control cycle. Values e_α and e_β (vector components of e_s) are used for the estimated Theta calculation.

FIGURE 18: BACK-EMF ESTIMATION MODEL



Back-EMF Filtering

To provide the filtering, a first-order, digital low-pass filter is used with Equation 4.

EQUATION 4: FIRST-ORDER DIGITAL LOW-PASS FILTER:

$$y(n) = y(n-1) + T2\pi f_c \bullet (x(n) - y(n))$$

To filter z to obtain e^*, 8kHz is substituted in the equation, which results in:

$$e(n) = e(n-1) + \left(\frac{1}{f_{pwm}}\right) \bullet 2\pi f_c (z(n) - e(n))$$

Where:

$e(n)$	=	Next estimated back-EMF value
$e(n-1)$	=	Last estimated back-EMF value
f_{pwm}	=	PWM frequency at which the digital filter is being calculated
f_c	=	Cut-off frequency of the filter
$z(n)$	=	Unfiltered back-EMF, which is output from the slide-mode controller

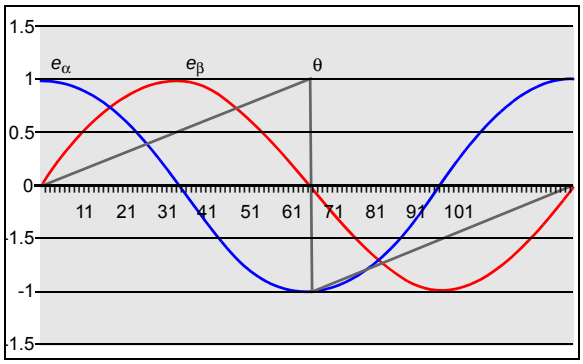
The cut-off frequency value is set to be equal to the frequency of the drive current and the motor voltage, which is the electrical revolutions per second. Due to the implementation of the adaptive filter, there is a fixed phase delay of -45° (per filter) for Theta compensation in all speed ranges, because the cut-off frequency changes as the motor gains speed.

The output of the first filter is used in two blocks. The first block is the model itself, used to calculate the next estimated current (i_s^*), and also to calculate the estimated Theta (θ^*). A second, first-order filter is used to calculate a smoother signal coming out of the motor model.

Relationship Between Back-EMF and Rotor Position

Once the back-EMF has been filtered for the second time, Theta is calculated. The relationship between e_s and θ can be explained based on the graph shown in Figure 19.

FIGURE 19: BACK-EMF AND THETA RELATIONSHIP



The plot shows a trigonometric function relating the vector components of the back-EMF (e_α and e_β) and rotor angle (θ). The arctangent is computed on the back-EMF vector components to calculate Theta. Equation 5 illustrates how the function is implemented in software:

EQUATION 5: THETA CALCULATION

$$\theta = arctan\ (e_\alpha, e_\beta)$$

The actual implementation uses a numeric and iterative algorithm called Coordinate Rotation by Digital Computer (CORDIC), which is fast, yet takes less memory than a floating point implementation. Discussion of the CORDIC algorithm is beyond the scope of this application note.

Speed Calculation

Due to the filtering function applied during the Theta calculation, some phase compensation is needed before the calculated angle is used to energize the motor windings. The amount of Theta compensation depends on the rate of change of Theta, or speed of the motor. The Theta compensation is performed in two steps:

1. The speed of the motor is calculated based on the uncompensated Theta calculation.
2. The calculated speed is filtered and used to calculate the amount of compensation, as shown in Figure 20.

Speed is calculated by accumulating Theta values over *m* samples and then multiplying the accumulated Theta by a constant. The formula used in this application note for speed calculation is shown in Equation 6.

EQUATION 6: SPEED CALCULATION

$$\omega = \sum_{i=0}^{m} (\theta_n - \theta_{n-1}) \bullet K_{speed}$$

Where,

Omega (ω)	=	Angular velocity of the motor
Theta (θn)	=	Current Theta value
PrevTheta (θn-1)	=	Previous Theta value
Kspeed	=	Amplification factor for desired speed range
m	=	Number of accumulated Theta deltas

To secure a smoother signal on the speed calculation, a first-order filter is applied to Omega (ω^*) to obtain FilteredOmega ($\omega^*_{filtered}$). The first-order filter topology is the same as the one used for back-EMF filtering.

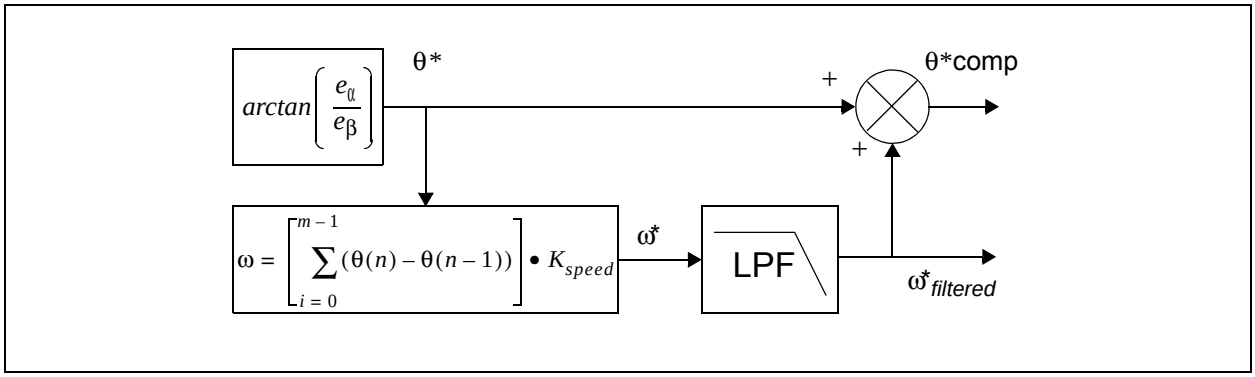
Adaptive Filter

The adaptive filter performs the following two functions:

- Calculation of gain for low-pass filter for the sliding-mode controller
- Dynamically compensate for Theta for the complete speed range

Two low-pass filters are implemented for the estimation of the position. The first filter, filters the output of sliding mode controller (correction factor (Z)) in to estimated back-EMF (es*) and the second filter, filters the estimated back-EMF (es*) into the filtered estimated back-EMF (efiltered*s). The derivation for calculating gain for low-pass filter is shown in Equation 7.

FIGURE 20: SPEED CALCULATION BLOCK DIAGRAM



EQUATION 7:

The following derivation explains the calculation of gain for low-pass filter, where:

Kslf	=	Gain of low-pass filter for sliding mode controller
Tpwm	=	Time period of PWM in seconds
eRPS	=	Electrical rotation of motor in RPS
Omega	=	Angular velocity of motor in rad/s
SpeedLoopTime	=	Time for speed loop execution in seconds
IRP_PERCALC	=	Number of PWM loops per speed loop

$$Kslf = Tpwm \bullet 2 \bullet PI \bullet eRPS \quad \text{--- (3)}$$

$$eRPS = (RPM \bullet Pole\_Pair)/60 \quad \text{--- (4)}$$

Also,

$$RPM = (Q15(Omega) \bullet 60)/(SpeedLoopTime \bullet Motor\ Poles) \quad \text{--- (5)}$$

Substituting **(5)** in **(4)**,

$$eRPS = (Q15(Omega) \bullet 60 \bullet Pole\ Pairs)/(SpeedLoopTime \bullet Pole\ Pairs \bullet 2 \bullet 60)$$

$$eRPS = Q15(Omega)/(SpeedLoopTime \bullet 2) \quad \text{--- (6)}$$

Substituting **(6)** in **(3)**,

$$Kslf = Tpwm \bullet 2 \bullet PI \bullet Q15(Omega)/(SpeedLoopTime \bullet 2) \quad \text{--- (7)}$$

Now,

$$IRP\_PERCALC = SpeedLoopTime/Tpwm \quad \text{--- (8)}$$

Substituting **(8)** in **(7)**,

$$Kslf = Tpwm \bullet 2 \bullet Q15(Omega) \bullet PI/(IRP\_PERCALC \bullet Tpwm \bullet 2)$$

Simplifying,

$$Kslf = Q15(Omega) \bullet PI/IRP\_PERCALC$$

We use a second filter to get a cleaner signal with the same coefficient:

$$Kslf = KslfFinal = Q15(Omega) \bullet PI/IRP\_PERCALC$$

The design of the adaptive filter keeps a fixed-phase delay for Theta compensation in all speed ranges as the cut-off frequency keeps changing with the increasing motor speed. Due to the implementation of two adaptive filters, the fixed phase delay of 90° is only a constant offset that is added to the calculated Theta.

FIELD WEAKENING

The field weakening for PMSM implies imposing a negative value for the stator current on the d-axis of the rotating frame, which has the role of weakening the air gap flux linkage.

In the case of an inverter, the voltage output drops on the stator's resistance and inductive reactance, and the remaining voltage is used to counteract BEMF. BEMF is proportional to the motor's speed and the voltage constant, KΦ, of the motor. Considering the inverter's limitation of maximum output voltage, an increase in speed (above nominal speed) can be achieved by decreasing the voltage constant (KΦ), which is proportional with the air gap flux linkage. However, a decrease in air gap flux linkage is synonymous to the decrease in torque. However, for certain applications, the motor needs to run higher than the rated speeds and therefore, the field weakening feature is useful, which increases the speed range of motor beyond its nominal speed rating.

In a PMSM, the field weakening is achieved by reducing the value of I_d from '0' to a negative value.

The field weakening table consists of values for I_d defined in `UserParms.h` (from `dqKFw0` to `dqKFw15`), which can be changed based on the user requirement.

If the user does not want to use the field weakening feature, allow `NOMINALSPEEDINRPM` to be equal to `FIELDWEAKSPEEDRPM` in `UserParms.h`.

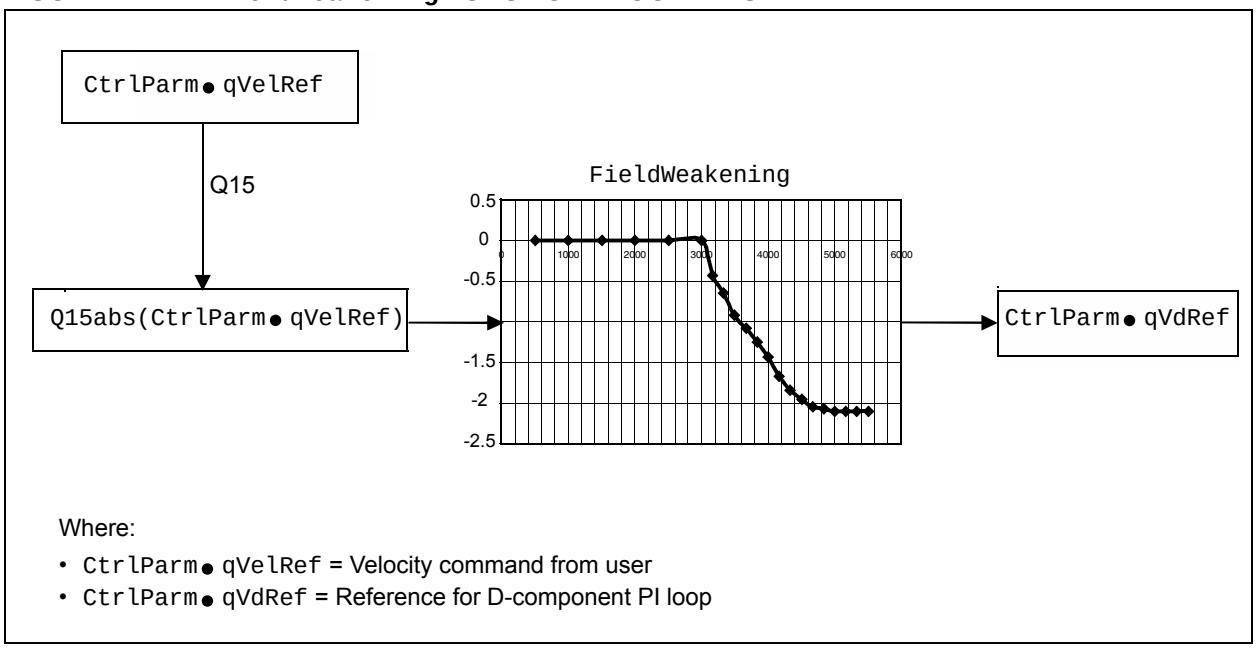
The `FieldWeakening` function takes the input parameter, velocity reference `CtrlParm●qVelRef`, and checks if the value is smaller or larger than `NOMINALSPEEDINRPM` defined in `UserParms.h`.

- If the velocity reference is smaller, field weakening is not performed
- If the velocity reference is larger, the suitable value is returned from `FdWeakParm.qFwCurve` through linear interpolation

Figure 21 illustrates the block diagram of field weakening.

Figure 22 illustrates a field weakening curve, used during testing.

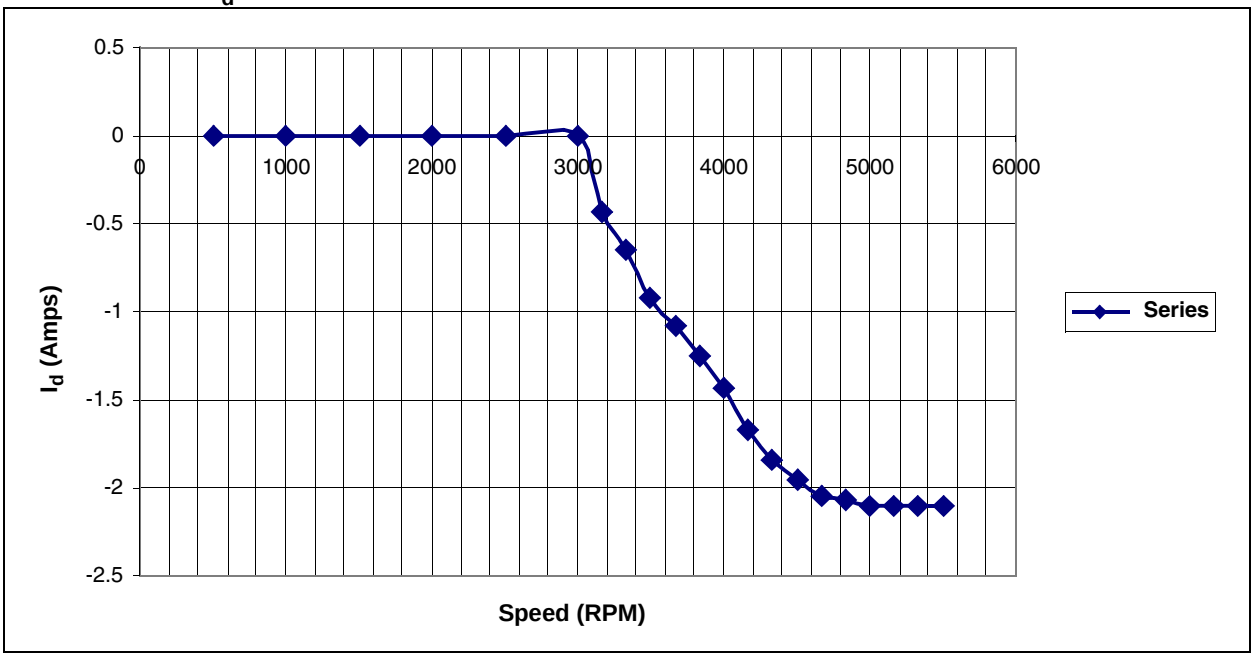
FIGURE 21: `FieldWeakening` FUNCTION BLOCK DIAGRAM



Where:

- `CtrlParm● qVelRef` = Velocity command from user
- `CtrlParm● qVdRef` = Reference for D-component PI loop

FIGURE 22: I_d vs. SHAFT RPM OF MOTOR



PERFORMANCE MODE

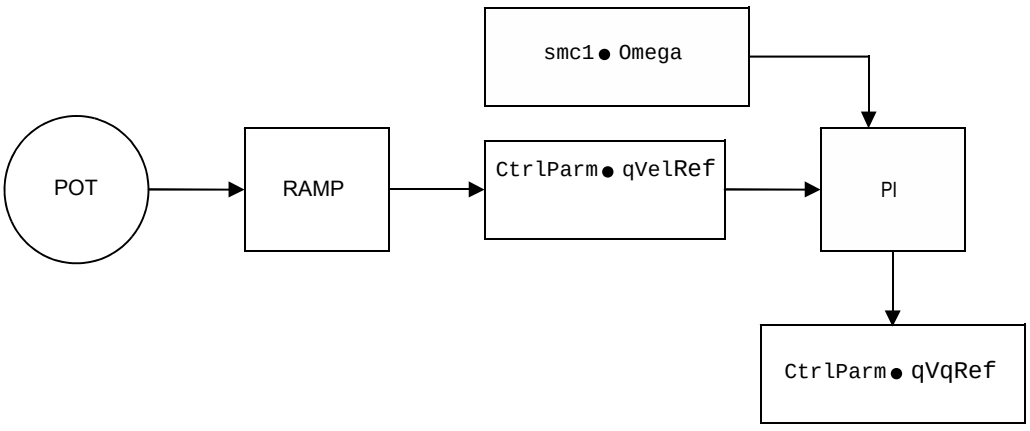
Speed Mode

In Speed mode, the measured motor velocity is compared with the reference from the potentiometer using a PI loop. The output of PI is applied as input to PI of Q-component of the current.

During the motor operation, the shaft RPM is held constant regardless of the variation in load. This mode can be selected by commenting the definition //#define TORQUEMODE in UserParms.h.

Figure 23 illustrates the block diagram of the Speed mode.

FIGURE 23: SPEED MODE BLOCK DIAGRAM



Where,

RAMP = Software module that provides a smooth acceleration and deceleration of motor based on input from potentiometer. To change the rate of RAMP, change SPEEDDELAY in UserParms.h.

smc1● Omega = Measured velocity of motor

CtrlParm● qVelRef = Velocity command from user

CtrlParm● qVqRef = Output of velocity PI loop and input for PI loop of Q-component

Torque Mode

In Torque mode, the velocity PI loop is bypassed and the reference from the potentiometer is directly fed as input to the PI loop of Q-component of current.

During the motor operation, the torque generated by the motor and the current consumption are held constant (as set by the potentiometer). Therefore, under a heavier load, the shaft RPM may drop. This mode can be selected by uncommenting the definition //#define TORQUEMODE in UserParms.h.

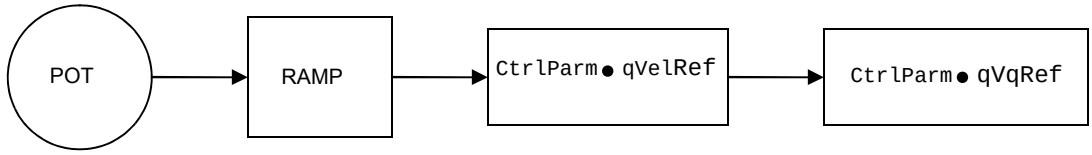
Figure 24 illustrates the block diagram of Torque mode.

Voltage Ripple Compensation

The ripple compensation is used to compensate the Vd and Vq (inputs to inverse park transform block) based on the DC bus ripple. The user can enable the ripple compensation section of the code through #define ENVOLTRIPPLE in UserParms.h. If this feature is enabled in the code, the software compensates for the voltage ripple on the DC bus. This in turn makes the hardware design economical by reducing the size of the buffer capacitor on the DC bus.

Equation 8 describes the implementation of ripple compensation for D-component. Figure 25 and Figure 26 illustrates the block diagram of bus voltage ripple compensation for D-axis and Q-axis, respectively.

FIGURE 24: TORQUE MODE BLOCK DIAGRAM



Where,

RAMP = Software module that provides a smooth acceleration and deceleration of motor based on input from potentiometer. To change the rate of ramp, change SPEEDDELAY in UserParms.h.

CtrlParm● qVelRef = Velocity command from user

CtrlParm● qVqRef = Output of velocity PI loop and input for PI loop of Q-component

EQUATION 8:

DCbus = Measured voltage of DC bus

TargetDCbus = Required voltage of DC bus

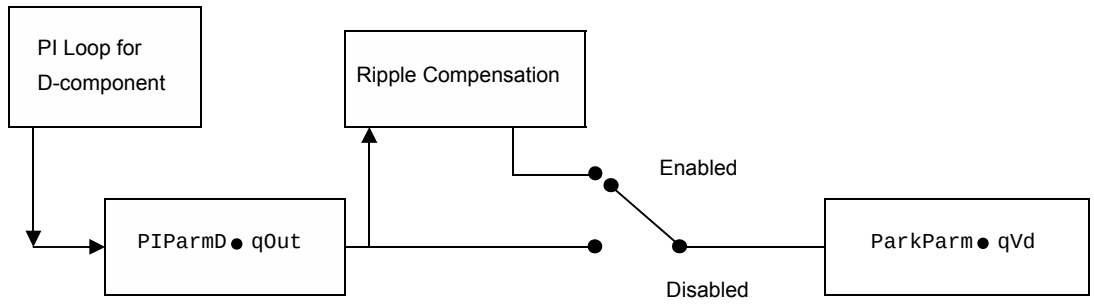
```
If(TargetDCbus > DCbus)
```

$$ParkParm \bullet qVd = PIParmD \bullet qOut + \frac{TargetDCbus - DCbus}{DCbus} \bullet PIParmD \bullet qOut$$

```
If(DCbus > TargetDCbus)
```

$$ParkParm \bullet qVd = \frac{TargetDCbus}{DCbus} \bullet PIParmD \bullet qOut$$

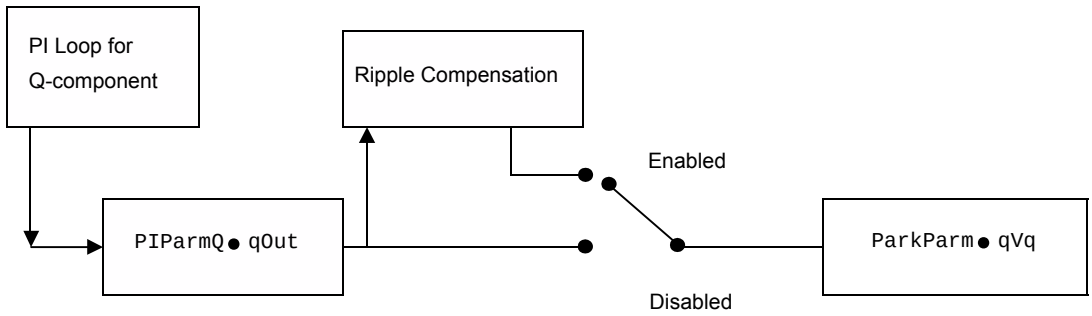
FIGURE 25: BUS VOLTAGE RIPPLE COMPENSATION FOR D-AXIS CURRENT(1,2)



Note 1: When voltage ripple compensation is disabled, the `PIParmD● qOut` (output of PI loop for D-component) is passed directly to `ParkParm● qVd`.

2: When voltage ripple compensation is enabled, the `PIParmD● qOut` is passed to the function `VoltRippleComp`. This function returns the compensated value which is passed to `ParkParm● qVd`.

FIGURE 26: BUS VOLTAGE RIPPLE COMPENSATION FOR Q-AXIS CURRENT(1,2)



Note 1: When voltage ripple compensation is disabled, the `PIParmQ● qOut` (output of PI loop for Q-component) is passed directly to `ParkParm● qVq`.

2: When voltage ripple compensation is enabled, the `PIParmQ● qOut` is passed to `VoltRippleComp`. This function returns the compensated value which is passed to `ParkParm● qVq`.

FLOW CHARTS

The FOC algorithm is executed at the same rate as the PWM. It is configured so that the PWM triggers A/D conversions for two windings using two shunt resistors and a potentiometer that sets the reference speed of the motor. Interrupts of the A/D are enabled to perform the algorithm. Figure 27 illustrates the general execution of the A/D interrupt subroutine.

FIGURE 27: A/D INTERRUPT SUBROUTINE

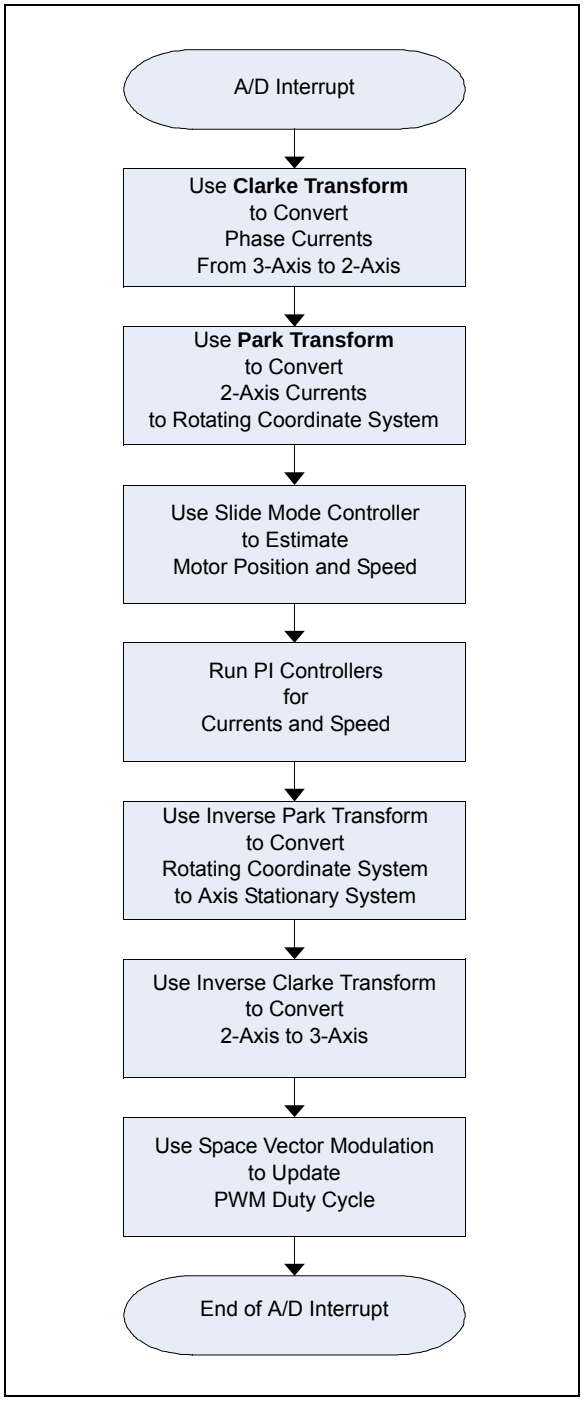
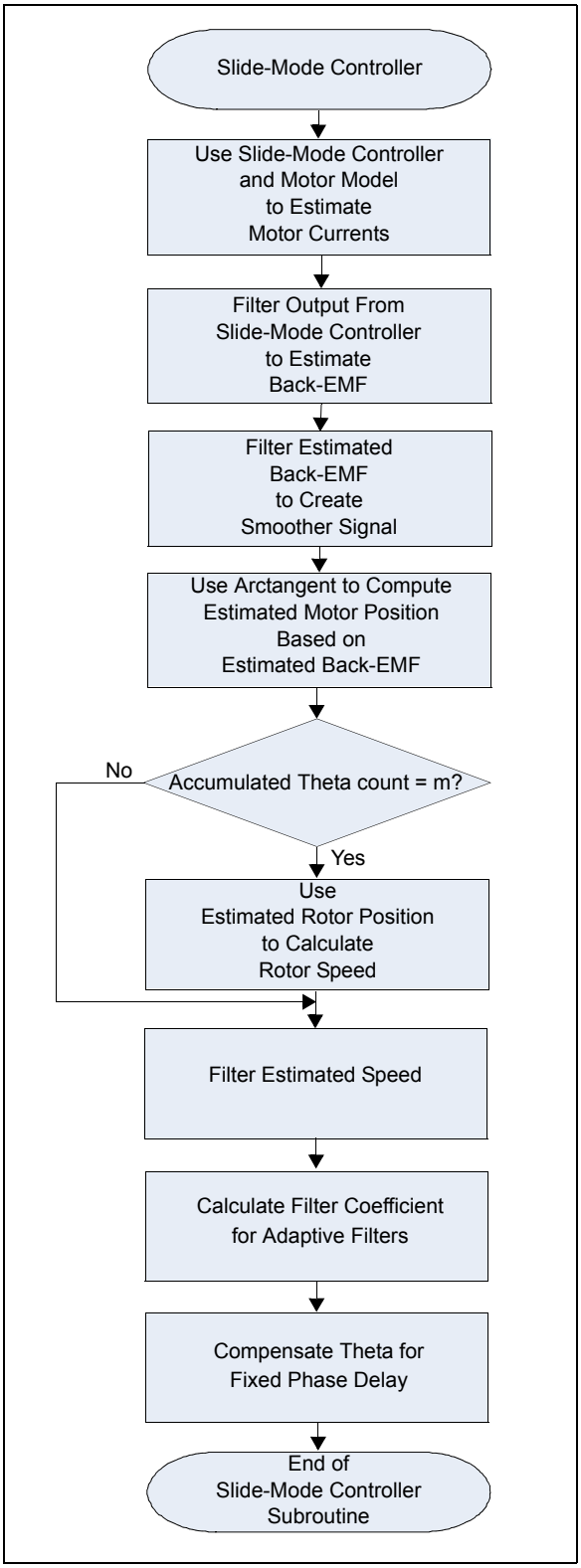


Figure 28 illustrates the process of using the Slide-Mode Controller to estimate the position and speed of the motor.

FIGURE 28: MOTOR POSITION AND SPEED ESTIMATION



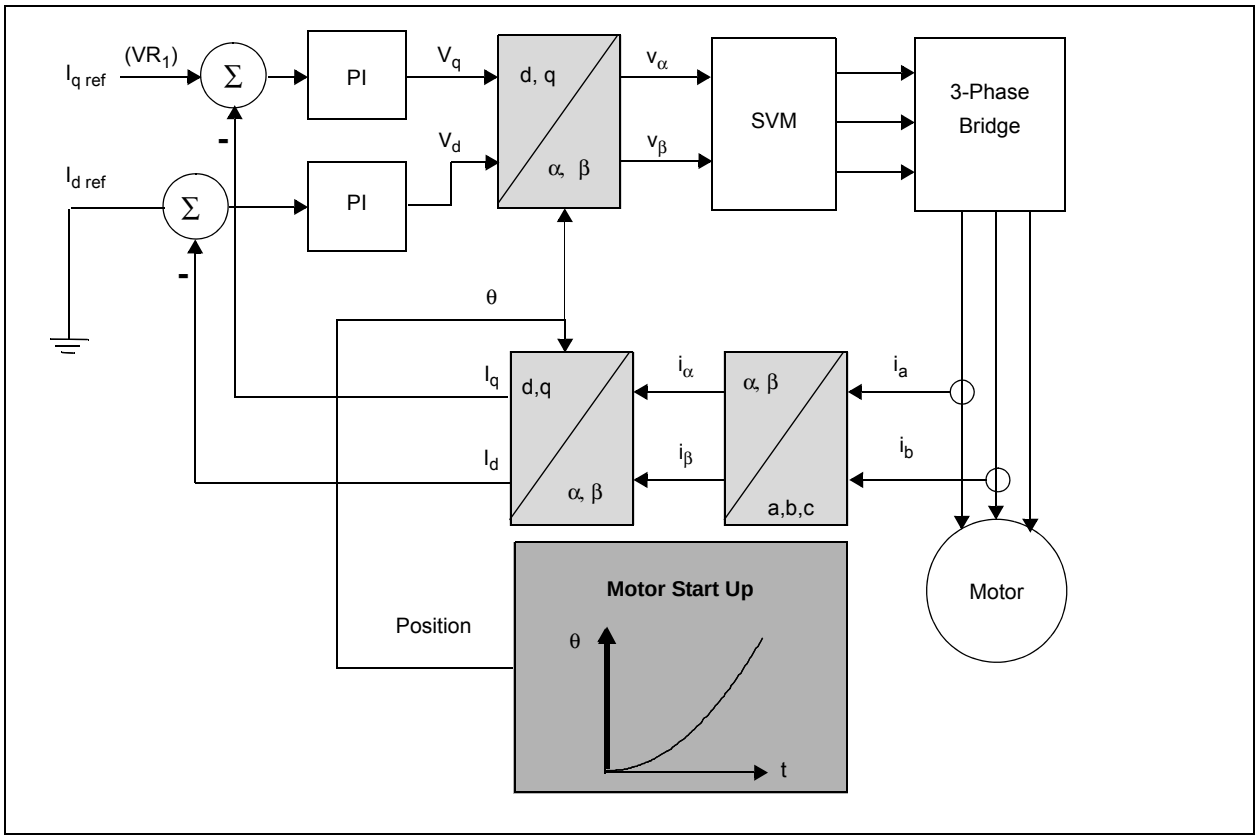
MOTOR START-UP

Since the sensorless FOC algorithm is based on the back-EMF estimation, a minimum speed is needed to get the estimated back-EMF value. Therefore, the motor windings must be energized with the appropriate estimated angle. To handle this, a motor start-up subroutine (see Figure 29) was developed.

When the motor is at a standstill and the Start/Stop button has been pressed, the dsPIC DSC generates a series of sinusoidal voltages to start the motor spinning. The motor spins at a fixed acceleration rate, and the FOC algorithm controls the currents I_d and I_q. The Theta angle (commutation angle) is incremented based on the acceleration rate.

As shown in the diagram, phase angle is incremented at a squared rate to get a constant acceleration on the motor. Even if Theta is being generated by the open-loop state machine, FOC blocks are still being executed and are controlling torque component current and flux component current. An external potentiometer is used to set the desired torque required to start the motor. This potentiometer is set experimentally depending on mechanical load characteristics. This start-up subroutine provides a constant torque to start up the motor. At the end of the start-up ramp, the software switches over to closed loop, sensorless control, taking Theta from the position and speed estimator, as shown in Figure 6.

FIGURE 29: MOTOR START-UP



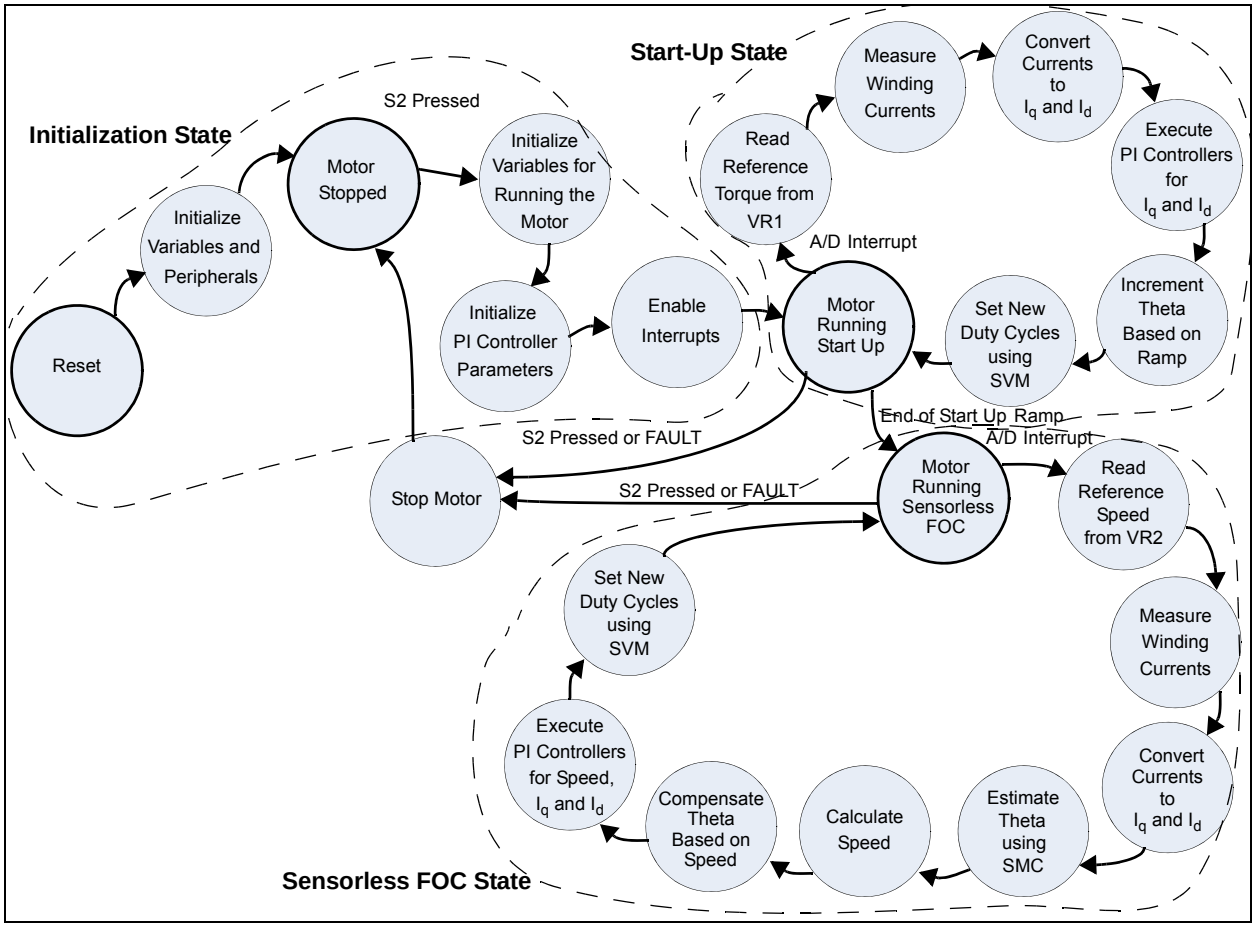
MAIN SOFTWARE STATE MACHINE

It is helpful to visualize the FOC algorithm as a software state machine (see Figure 30). First, the motor windings are de-energized and the system waits for the user to press the Start/Stop button (S2 on the dsPICDEM MCLV Development Board). Once the user presses Start/Stop button, the system enters the initialization state, where all variables are set to their initial value and interrupts are enabled. Then, the start-up subroutine is executed, where current components for torque (I_q) and flux production (I_d) are being controlled, and commutation angle (Theta) is being generated in a ramp fashion to get the motor spinning.

After going through the start-up subroutine, the system switches over to sensorless FOC control, where the speed controller is added to the execution thread, and the Slide-Mode Controller (SMC) starts estimating Theta as previously explained. When the motor enters sensorless FOC control state, the reference speed is continuously read from an external potentiometer and the start/stop button is monitored to stop the motor.

Any fault in the system causes the motor to stop and return to the Motor Stopped state until S2 is pressed again. The state diagram shows all the different states of the software and the conditions that make the system transition to a different state.

FIGURE 30: MAIN SOFTWARE STATE MACHINE



BENEFITS OF DSC-BASED FOC CONTROL

A major advantage of deploying DSCs in motor control is the practicality of a common design platform, which makes the production of appliances more efficient. This means appliance makers now have an economical way to offer a range of appliance models that use PMSM or other type of motors with sensorless FOC algorithm control.

These software-based motor control designs enable rapid customization to address multiple markets by changing only the control parameters.

Protection of firmware Intellectual Property (IP) is another major issue for manufacturers who frequently deploy appliance design teams that collaborate across many geographies. It is easy to imagine a scenario in which the implementation of FOC for an appliance may have come from place A, the user-interface board from place B and the final system integration being done in place C.

CONCLUSION

This application note illustrates how designers can take advantage of DSCs to implement advanced motor control techniques such as the Sensorless FOC algorithm in appliance applications. Since programming a dsPIC DSC is similar to programming a MCU, appliance designers can quickly design their motor control algorithm and test their prototypes.

Fine tuning motor control is made easy, thanks to the powerful IDE-based tools such as the DMCI, which allow the designer to easily port their algorithms across a variety of motor platforms including PMSM, BLDC, BDC and ACIM.

REFERENCES

Several application notes have been published by Microchip Technology describing the use of DSCs for motor control.

For ACIM control see:

- *AN984*, *"An Introduction to AC Induction Motor Control Using the dsPIC30F MCU"* (DS00984)
- *AN908*, *"Using the dsPIC30F for Vector Control of an ACIM"* (DS00908)
- *GS004*, *"Driving an ACIM with the dsPIC® DSC MCPWM Module"* (DS93004)
- *AN1206, "Sensorless Field Oriented Control (FOC) of an AC Induction Motor (ACIM) Using Field Weakening"* (DS01206)
- *AN1162, "Sensorless Field Oriented Control (FOC) of an AC Induction Motor (ACIM)"* (DS01162)

For BLDC motor control see:

- *AN957*, *"Sensored BLDC Motor Control Using dsPIC30F2010"* (DS00957)
- *AN1160, "Sensorless BLDC Control with Back-EMF Filtering Using a Majority Function"* (DS01160)

For PMSM control see:

- *AN1017*, *"Sinusoidal Control of PMSM Motors with dsPIC30F DSC"* (DS01017)
- *AN1292, "Sensorless Field Oriented Control (FOC) for a Permanent Magnet Synchronous Motor (PMSM) Using a PLL Estimator and Field Weakening (FW)"* (DS01292)
- *AN1299, "Single-Shunt Three-Phase Current Reconstruction Algorithm for Sensorless FOC of a PMSM"* (DS01299)

For Stepper motor control see:

- *AN1307, "Stepper Motor Control with dsPIC® DSCs"* (DS01307)

For information on the dsPICDEM MC1 Motor Control Development Board see:

- *"dsPICDEM™ MCLV Development Board User's Guide"* (DS70331)
- *"dsPICDEM™ MCHV Development System User's Guide*" (DS70605)
- *"dsPICDEM™ MCSM Development Board User's Guide*" (DS70610)
- *"dsPICDEM™ MC1 Motor Control Development Board User's Guide"* (DS70098)
- *"dsPICDEM™ MC1H 3-Phase High Voltage Power Module User's Guide"* (DS70096)
- *"dsPICDEM™ MC1L 3-Phase Low Voltage Power Module User's Guide"* (DS70097)

These documents are available on the Microchip web site (www.microchip.com).

APPENDIX A: HARDWARE RESOURCES

The sensorless FOC code for a PMSM has been tested on the following development boards:

- dsPICDEM™ MCLV Development Board (DM330021)
- dsPICDEM™ MCHV Development Board (DM330023)

REVISION HISTORY

Revision A (March 2007)

This is the initial release of this document.

Revision B (March 2010)

This revision incorporates the following updates:

- Updated third paragraph in **"System Overview"**:
- Added the following sections:
  - **"Field Weakening"**
  - **"Performance Mode"**
- Note:
  - Added a note with information regarding the use of asterik (*) as an estimated variable (see note above Figure 17).
- Sections:
  - Updated the speed range from 500RPM-7300 RPM to 500RPM-17000 RPM in **"Application Highlights"**.
  - Removed the following point in **"Application Highlights"**: An air conditioner compresser rated at 1.5 kw is targeted as the main motor.
  - Removed Phase Compensation, Changing Phase Compensation Formulas section.
  - Added a sub section **"Adaptive Filter"**.
  - Added a sub section **"Voltage Ripple Compensation"**.
- Appendix :
  - Removed Hardware Modifications section in **Appendix A: "Hardware Resources"**.
- Figures:
  - Removed Figure 25 through Figure 30 from **Appendix A: "Hardware Resources"**.
- Additional minor corrections such as language and formatting updates are incorporated throughout the document.

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

ISBN: 978-1-60932-099-7

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://support.microchip.com
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-6578-300
Fax: 886-3-6578-370

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/05/10