

Sensorless BLDC Control With Back-EMF Filtering

*Author: Reston Condit
Microchip Technology Inc.*

INTRODUCTION

This application note describes a sensorless brushless DC (BLDC) motor control algorithm, implemented using the dsPIC® digital signal controller (DSC). The algorithm works by digitally filtering the back-EMF on each phase of the motor and determining when to commutate the motor windings based on the filtered back-EMF signals. This control technique precludes the need for discrete, low-pass filtering hardware and off-chip comparators.

BLDC motors are used in a variety of applications. The algorithm described in this application note targets BLDC motors that operate in the 40k to 100k electrical RPM range. Some BLDC motor applications that run in this RPM range are model RC motors, fans, hard drives, air pumps and dental drills.

The algorithm described in this application note can be implemented on two Microchip development board platforms:

- PICDEM™ MC LV Development Board
- dsPICDEM™ MC1 Development Board

The PICDEM™ MC LV Development Board includes a dsPIC30F3010 DSC. The described algorithm was implemented on this device because it is included with the PICDEM™ MC LV Development Board. However, for cost reduction, you can use the dsPIC30F2010 as a substitute processor.

In its default configuration, this board includes a 5 MHz crystal. A 7.37 MHz crystal was used during the testing of this algorithm.

The resources used on the PICDEM MC LV Development Board are:

| | |
|-----------------|-------------------------------|
| Processor Type: | dsPIC30F3010 or dsPIC2010 |
| MIPS: | 21 MIPS |
| Program Memory: | 2000 24-bit instruction words |
| RAM: | 280 bytes |
| Crystal: | 7.68 MHz |

The resources used on the dsPICDEM MC1 Development Board are:

| | |
|-----------------|-------------------------------|
| Processor: | dsPIC30F6010A |
| MIPS: | 21 MIPS |
| Program Memory: | 2089 24-bit instruction words |
| RAM: | 280 bytes ⁽¹⁾ |

Note 1: With signal buffers enabled, RAM usage is 4400 bytes.

These specifications are common to both hardware platforms:

- Maximum Motor Speed: 100,000 Electrical RPM
- Tunable PID Speed Control Loop
- Configurable Open-Loop Start-up Ramp
- Supports the DMCI Tool (see the “**Implementing the Algorithm**” section of this document)

FIGURE 1: PICDEM™ MC LV DEVELOPMENT BOARD

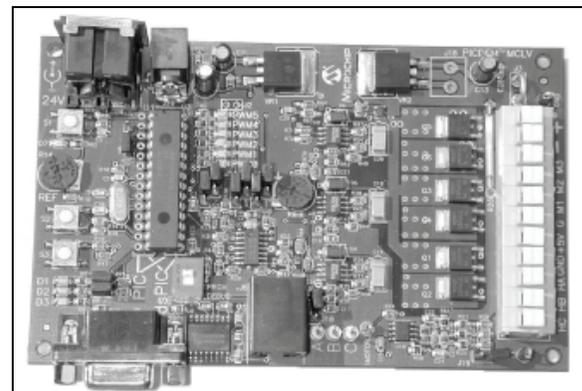


FIGURE 2: dsPICDEM™ MC1 DEVELOPMENT BOARD WITH ATTACHED POWER MODULE



BLDC MOTOR CONSTRUCTION

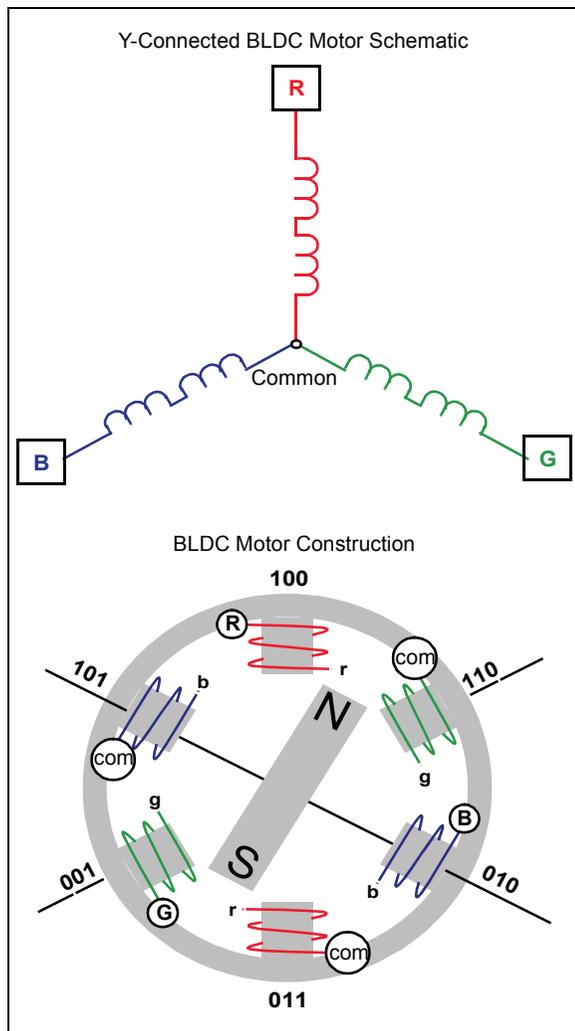
This algorithm has been tested on these motors:

- 4-pole, 12V, 10A fan motor with a maximum speed of 29,000 RPM
- 14-pole, 12V model airplane motor with a maximum speed of 13,000 RPM
- 4-pole, 24V, 1A Hurst BLDC motor model DMB0224C10002

These motors have been tested with loads that are proportional to speed – they have light loads at low speed and full loads at high speed.

In the Y-connected BLDC motor (see Figure 3), the motor has three leads, each connected to a winding. Each winding (or series of windings) is connected to a common point shared by all three windings, as shown in the upper portion of Figure 3. The basic construction of a simple BLDC motor is shown in the lower portion of Figure 3.

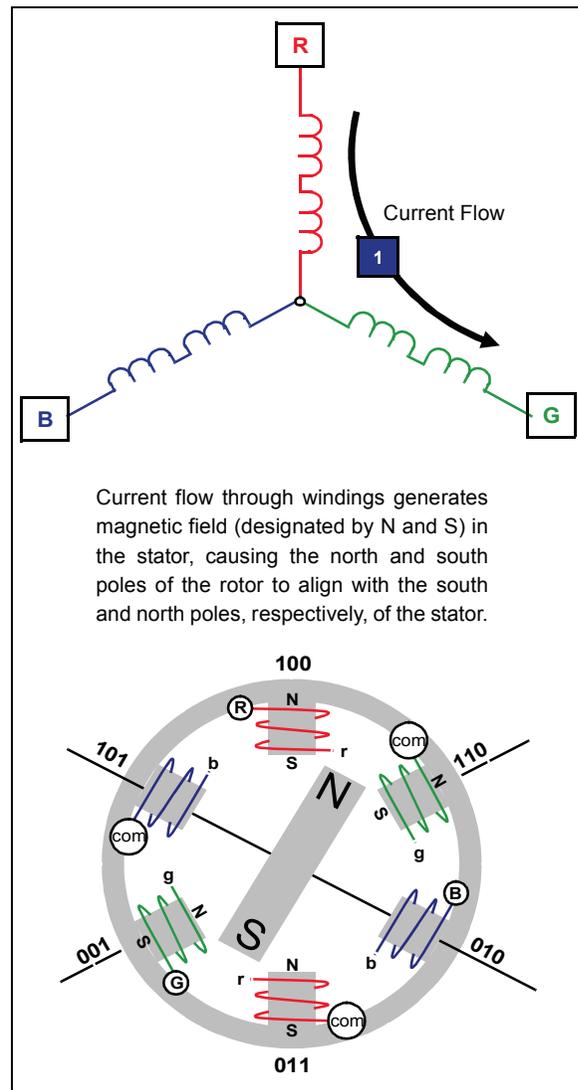
FIGURE 3: Y-CONNECTED BLDC MOTOR



The outer layer of the motor, which houses the motor windings, is the stator. The inner part of the motor is the rotor. The rotor consists of opposing magnetic poles located around the circumference of the motor. Figure 4 shows a rotor with only two poles, north and south. In reality, most motors have more than one set of magnetic poles on the rotor.

The motor turns when a current is passed through motor windings, as shown by the arrow (1) in Figure 4. In this example, a positive potential is applied to the red (R) lead, and a negative potential is applied to the green (G) lead. Charging the motor windings in this manner generates the magnetic field in the stator, noted by the N and S designations. The rotor then turns so that the north pole in the rotor aligns with the magnetic south generated in the stator. Likewise, the south pole in the rotor aligns with the magnetic north generated in the stator.

FIGURE 4: CURRENT FLOW THROUGH WINDINGS

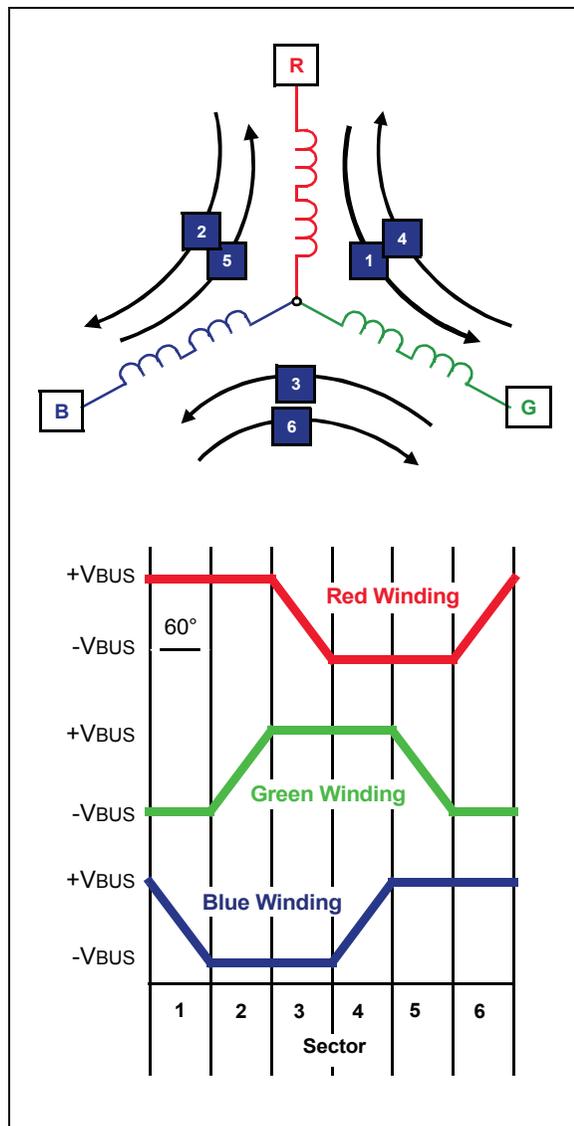


SIX-STEP (TRAPEZOIDAL) COMMUTATION

The method for energizing the motor windings in the sensorless algorithm described in this application note is six-step trapezoidal or 120° commutation. Figure 5 shows how six-step commutation works. Each step, or sector, is equivalent to 60 electrical degrees. Six sectors make up 360 electrical degrees, or one electrical revolution.

The arrows in the winding diagram show the direction current flows through the motor windings in each of the six steps. The graph shows the potential applied at each lead of the motor during the six steps. Sequencing through these six steps moves the motor one electrical revolution.

FIGURE 5: SIX-STEP COMMUTATION



| Step | Commutation |
|------|---|
| 1 | Red winding is driven positive. Green winding is driven negative. Blue winding is not driven. |
| 2 | Red winding remains positive. Blue winding is driven negative. Green winding is not driven. |
| 3 | Green winding is driven positive. Blue winding is driven negative. Red winding is not driven. |
| 4 | Green winding is driven positive. Red winding is driven negative. Blue winding is not driven. |
| 5 | Blue winding is driven positive. Red winding is driven negative. Green winding is not driven. |
| 6 | Blue winding is driven positive. Green winding is driven negative. Red winding is not driven. |

For every sector, two windings are energized and one winding is not energized. The fact that one of the windings is not energized during each sector is an important characteristic of six-step control that allows for the use of a sensorless control algorithm.

This application note uses these terms to describe motor speed:

- Electrical revolutions per minute (RPM_{Elec})
- Electrical revolutions per second (RPS_{Elec})

It is easier to discuss motor speed in these terms rather than mechanical RPM because when talking about electrical RPM, the number of motor poles does not have to be factored in. The relationship between mechanical and electrical RPM is seen in these equations:

EQUATION 1: MECHANICAL/ELECTRICAL RPM RELATIONSHIP

$$RPM_{Mech} = \frac{(2 * RPM_{Elec})}{(\text{No. of Motor Poles})}$$

$$RPM_{Elec} = \frac{(RPM_{Mech} * \text{No. of Motor Poles})}{2}$$

$$RPS_{Elec} = \frac{RPM_{Elec}}{60}$$

To keep the magnetic field in the stator advancing ahead of the rotor, the transition from one sector to another must occur at precise rotor positions for optimal torque. The next section discusses how rotor position is determined in a sensed BLDC control application.

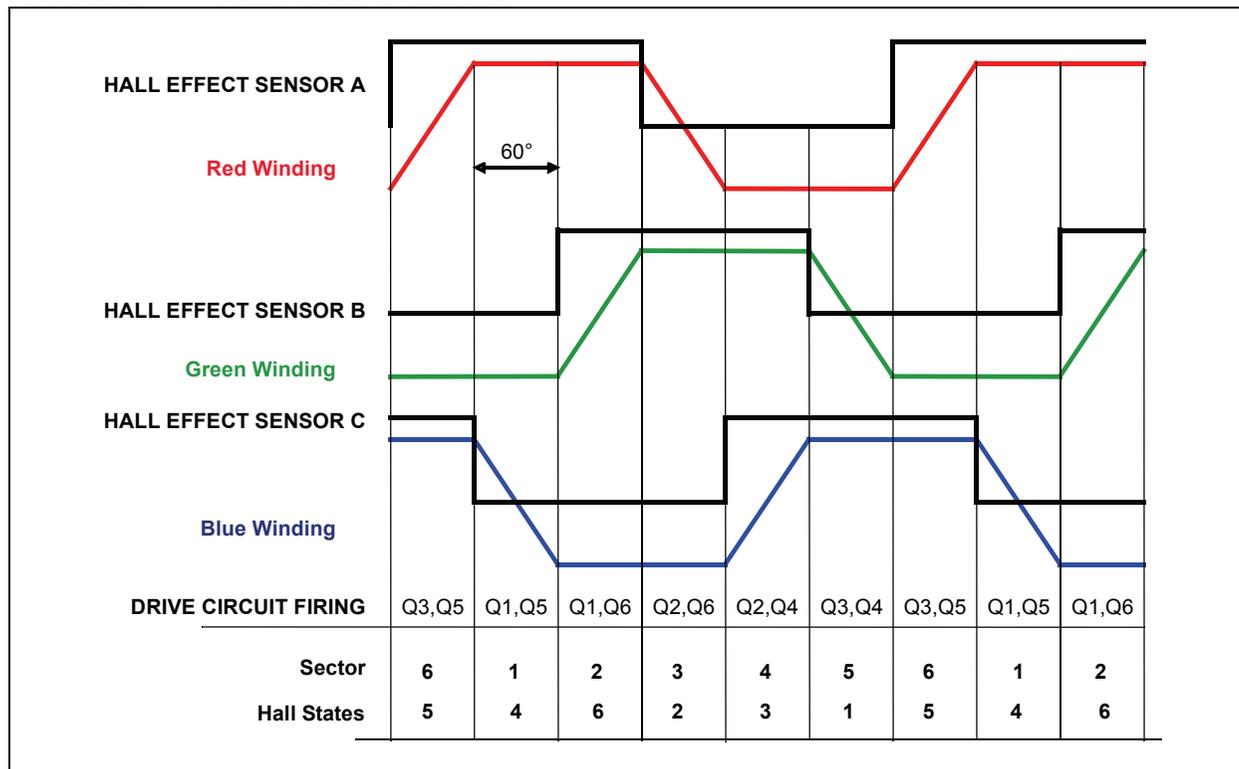
AN1083

SENSORED BLDC CONTROL

In most sensed BLDC control applications, Hall effect sensors are used to determine rotor position. Hall effect sensors are positioned in the motor housing in such a way that each sensor output changes state every

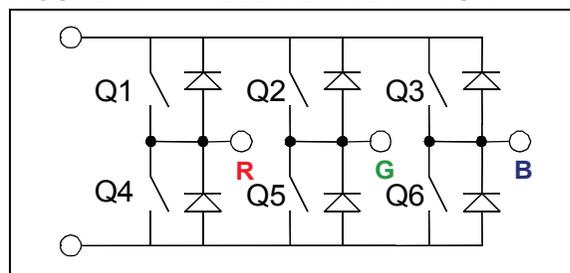
180 electrical degrees (see Figure 6). The rising edge of Sensor B is offset by 120 electrical degrees with respect to Sensor A. The rising edge of Sensor C is offset by 120 electrical degrees with respect to Sensor B. The sensors are positioned so that they change state when it is time for motor commutation to occur.

FIGURE 6: SENSORED CONTROL



A basic drive circuit for a BLDC motor is shown in Figure 7. Each motor lead is connected to a high-side and a low-side switch. The correlation between the sector and the switch states is noted by the drive circuit firing shown in Figure 6.

FIGURE 7: BLDC DRIVE CIRCUIT



WHY SENSORLESS CONTROL?

In sensed control, sensors determine the position of the motor rotor with respect to the motor stator. This makes for fairly simple control of the motor. A processor need only wait for a Hall effect sensor to change state, determine which sector the rotor is in based on the output of the three Hall effect sensors and commutate the motor windings appropriately.

Sensored control has several drawbacks:

- Sensors cost money. In addition to the sensor itself, there is the further cost of mounting the sensors to the motor during manufacturing as well as the cost of sensor wires.
- Sensors add another potential failure point to the motor. If a sensor fails, the motor fails.
- In some environments it is just not practical to use sensors. For instance, in an environment where the motor is flooded (like a compressor or pump), the sensors may be subject to failure before the rest of the motor.

For these and other reasons, sensorless BLDC control is desirable in many applications. The next section introduces the theory behind the sensorless BLDC control technique described in this application note.

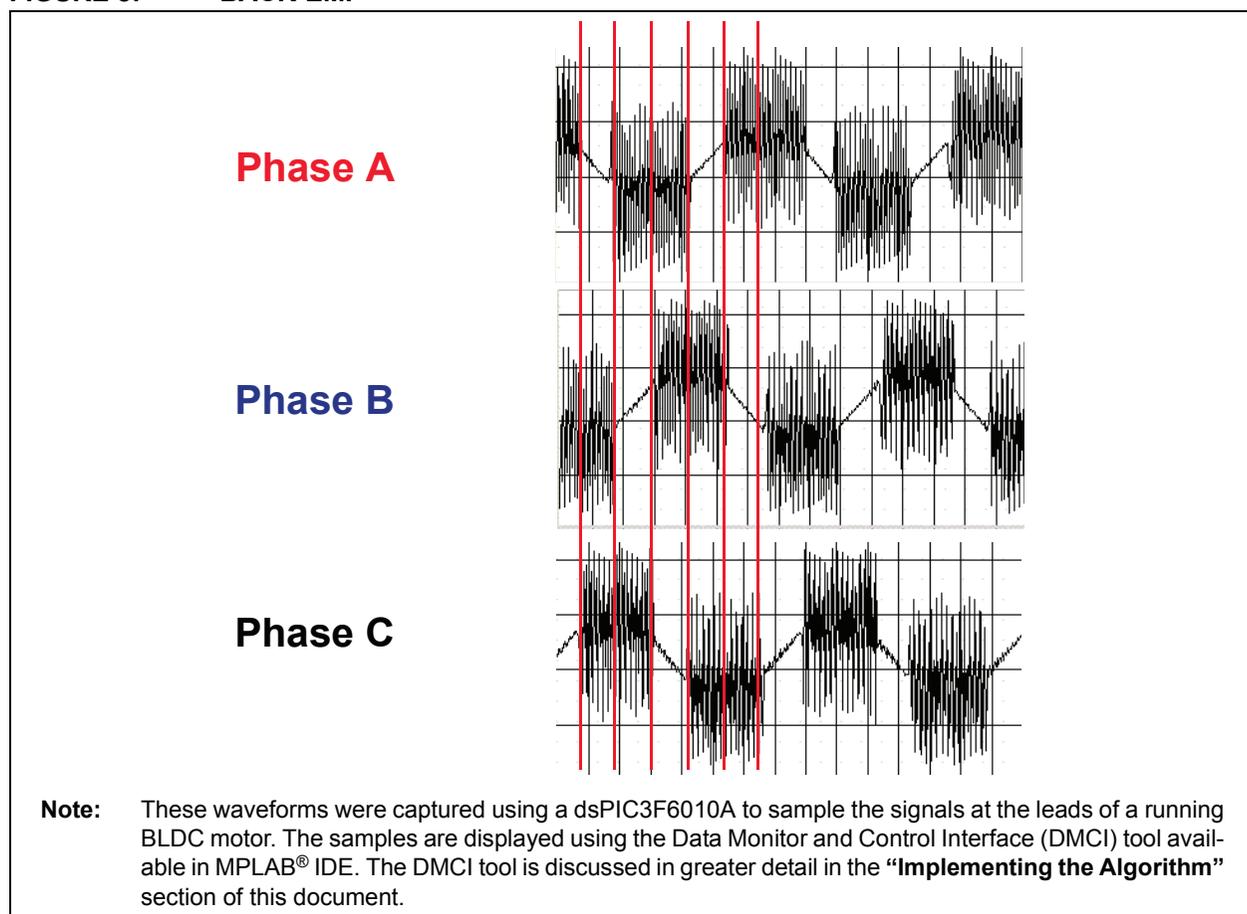
SENSORLESS BLDC CONTROL

The key to determining the appropriate moment when the motor winding should be commutated, or transition from one sector to another, is the rotor position. With sensed control, the rotor position is immediately evident. However, in sensorless control, rotor position must be determined by a means other than sensors.

The method chosen for this algorithm is to analyze the back-EMF from the motor. In simplest terms, back-EMF is the voltage generated in the stator winding by a permanent magnet motor when the rotor of the motor is turning. The magnitude of back-EMF is proportional to the speed of the motor.

Figure 8 will help to explain back-EMF. This figure shows the potential of the three leads of a motor as it turns using sensed control. The shape of the three signals is very similar to the trapezoidal shape shown in Figure 5. Vertical lines have been added to the graph to help identify the six sectors. A PWM signal is providing power to each winding. During every third sector, Phase A, for example, is not being driven. However, some voltage is still seen at the lead of the motor during the non-driven sector for a particular phase. This voltage is the back-EMF.

FIGURE 8: BACK-EMF



AN1083

The back-EMF signal during the undriven sector is actually being generated by the magnetized rotor signal moving past the windings in the stator. The back-EMF signal has three key characteristics that are important to sensorless control. These are:

- As speed increases, the magnitude of the back-EMF signal increases.
- As speed increases, the slope of the back-EMF signal becomes greater.
- The back-EMF signal is symmetrical around 0V (assuming the drive rails are plus/minus some voltage).

The third characteristic is shown in greater detail in Figure 9, which shows the back-EMF signal during the non-driven sector for Phase A. In this example, the motor is driven by $\pm 12V$. The back-EMF signal is symmetrical about 0V. (If the drive rails were 12V and 0V, the signal would be symmetrical at about 6V.) If the back-EMF signal was a straight line, the signal would cross the 0V line halfway through the sector, or at 30 electrical degrees, into the sector. This point is called the zero-cross event. The zero-cross event always occurs 30 degrees before the next commutation should occur. Therefore, given an algorithm that can

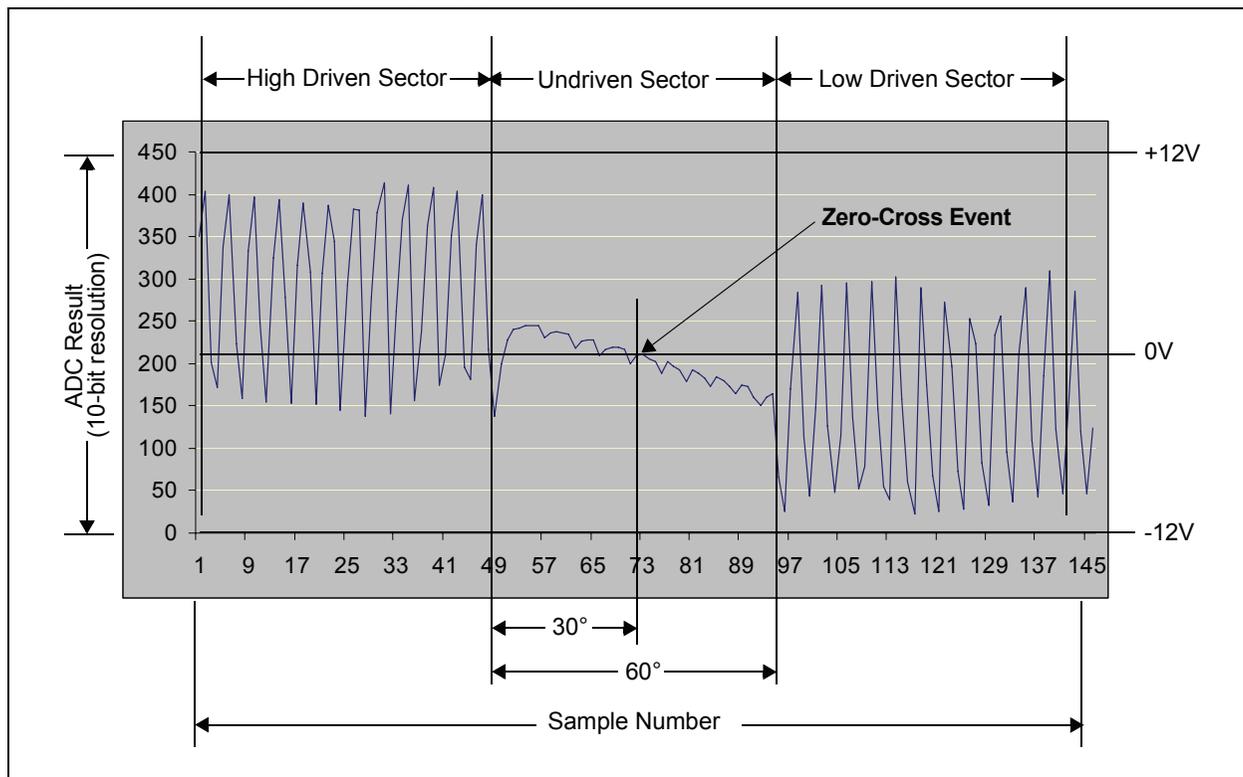
identify the zero-cross event accurately, the rotor position can be estimated and the motor windings can be commutated at the correct time.

Ideally, the undriven sector back-EMF signal would be a straight line. In actuality, the back-EMF signal has noise from the driven sectors coupled onto the signal. A PWM signal is used to vary the voltage to, and therefore, the speed of the motor. Because the motor windings are in proximity to one another, the PWM drive signal from one winding is coupled onto the back-EMF signal of another winding.

It is difficult to detect the zero-cross event due to the coupled PWM noise. For instance, if a microcontroller is instructed to identify the first time the back-EMF signal crosses the 0V threshold in the undriven sector for a particular phase, the microcontroller does not detect the crossing at 30 electrical degrees into the sector because PWM noise causes the signal to cross the 0V threshold prematurely. In Figure 9, it is apparent that the back-EMF signal crosses the 0V line two times before the 30-degree mark.

Detecting a zero-cross event accurately is the key to implementing this sensorless algorithm.

FIGURE 9: ZERO-CROSSING VOLTAGE



DIGITAL FILTERING

To reiterate, the back-EMF signal is not a clean signal. Coupled PWM noise from the other phases makes it difficult to accurately detect a zero-cross event. Figure 10 shows an actual back-EMF signal with an ideal signal transposed on top of it.

By using the dsPIC® DSC to implement a digital filter, a filtered back-EMF signal can be generated that looks more like the ideal signal. This is the premise for the

algorithm described in this application note – given a filtered back-EMF signal, it is easier to detect an actual zero-cross event (30 electrical degree mark) more accurately. When a zero-cross event is detected, the dsPIC DSC can then set up a timer to count down until it is time to commutate the motor windings.

Filtering a signal does have one drawback, however. Whether done digitally or in hardware, a filtered signal exhibits some phase delay as compared to the actual signal. The phase delay is illustrated in Figure 11.

FIGURE 10: ACTUAL AND IDEAL BACK-EMF SIGNAL

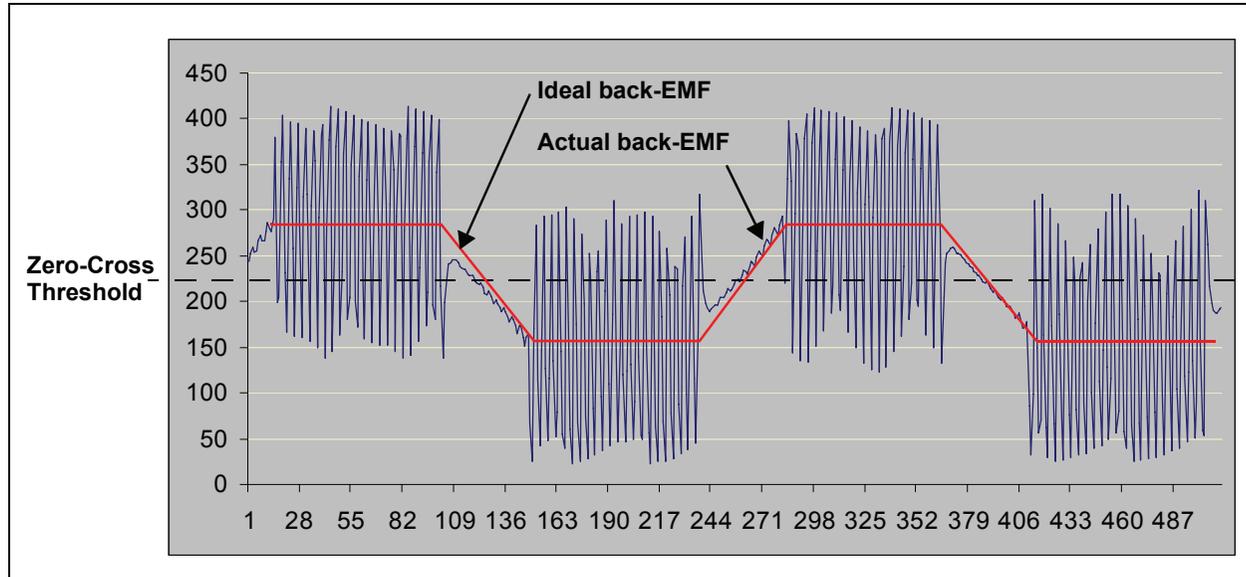
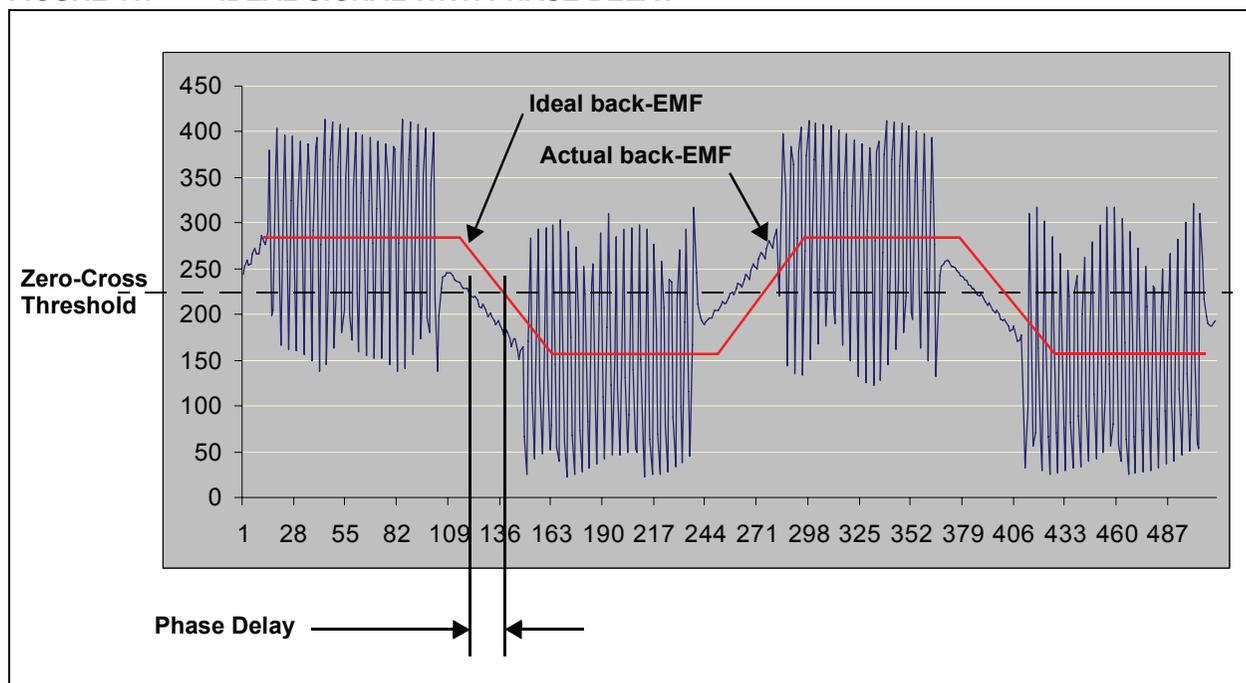


FIGURE 11: IDEAL SIGNAL WITH PHASE DELAY



Fortunately, when a digital filter is used, the phase delay can be calculated given the filter specifications and the frequency of the back-EMF signal. The calculated phase delay can then be subtracted from the count-down timer.

For this algorithm, the Microchip dsPIC DSC Digital Filter Design tool was used to design the back-EMF filter. This design tool automatically calculates phase delay for the filter. The next few paragraphs describe how the dsPIC DSC Digital Filter Design tool was used to design a filter for the 20 kHz PWM signal.

The Digital Filter Design tool requires the following inputs:

- Filter Type
- Sampling Frequency
- Pass-Band Frequency
- Stop-Band Frequency
- Pass-Band Ripple (db)
- Stop-Band Ripple (db)
- Filter Order

For this implementation, an IIR filter was chosen instead of an FIR filter because an IIR filter takes less time to execute. An FIR filter requires a much higher order when compared to an IIR filter with similar response characteristics. In addition, the phase delay period is much less for an IIR filter versus a FIR filter with a similar response.

The filter parameters used in this algorithm are listed in Table 1.

TABLE 1: FILTER PARAMETERS

| Parameter | Value |
|---------------------|--------------------------|
| Filter Type | Low-Pass Butterworth IIR |
| Sampling Frequency | 49152 Hz |
| Pass-Band Frequency | 4000 Hz |
| Stop-Band Frequency | 8000 Hz |
| Pass-Band Ripple | 0.1 db |
| Stop-Band Ripple | 60 db |
| Filter Order | 5th |

Note: Two filters are actually used in the sensorless BLDC algorithm. The reason for the two filters and their specifications is covered later in the document.

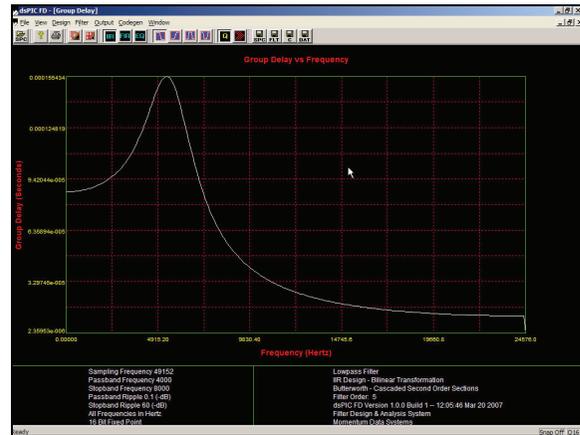
Once these parameters are specified, the dsPIC DSC Digital Filter Design tool calculates the IIR filter coefficients and generates the necessary source and header files for implementing the filter. These files have been added to the sensorless BLDC motor control project in MPLAB® IDE.

The dsPIC DSC Digital Filter Design tool provides several graphs showing the response of the filter. These graphs include:

- Magnitude vs. Frequency
- Log Magnitude vs. Frequency
- Phase vs. Frequency
- Group Delay vs. Frequency
- Impulse Response vs. Time
- Step Response vs. Time

Of special interest to the algorithm described in this application note is the Group Delay vs. Frequency graph. This graph is shown in Figure 12.

FIGURE 12: GROUP DELAY PLOT



The group delay plot shows how much delay (in seconds) the filtered signal will have when compared to the actual back-EMF signal for a given electrical revolution frequency. Assuming the maximum motor speed of 100,000 electrical RPM, the maximum electrical revolutions per second are 1666. According to the group delay plot, at 1666 Hz, the filtered signal will have a phase delay of approximately 90 μ s. At 0 Hz, the phase delay is approximately 87 μ s. The difference in delay time is negligible, so the phase delay can be treated as a constant over the entire speed range of the motor.

The next section explains how phase delay is factored into the sensorless BLDC motor control algorithm.

CHARACTERISTICS OF BACK-EMF

The characteristics of the back-EMF signal are quite different when the motor is turning slowly versus turning fast. As a result, the sensorless BLDC control algorithm analyzes the back-EMF in one of two ways, based on the speed of the motor. To understand why there are two implementations, it is necessary to look at the characteristics of the back-EMF signal at both low and high motor speeds.

Low-Speed back-EMF Characteristics

At very low speed, no noticeable back-EMF is generated. The back-EMF signal is essentially flat during the non-driven sector for a particular phase. Consequently, the zero-cross point is indistinguishable. As the motor turns faster, the back-EMF signal eventually begins to have some slope.

A sensorless algorithm will not work unless there is sufficient slope in the back-EMF signal to allow the zero-cross event to be determined accurately. For this reason, all sensorless BLDC algorithms must have an open-loop start-up ramp before the sensorless algorithm is able to take over and commutate the motor windings. During the open-loop start-up ramp, the

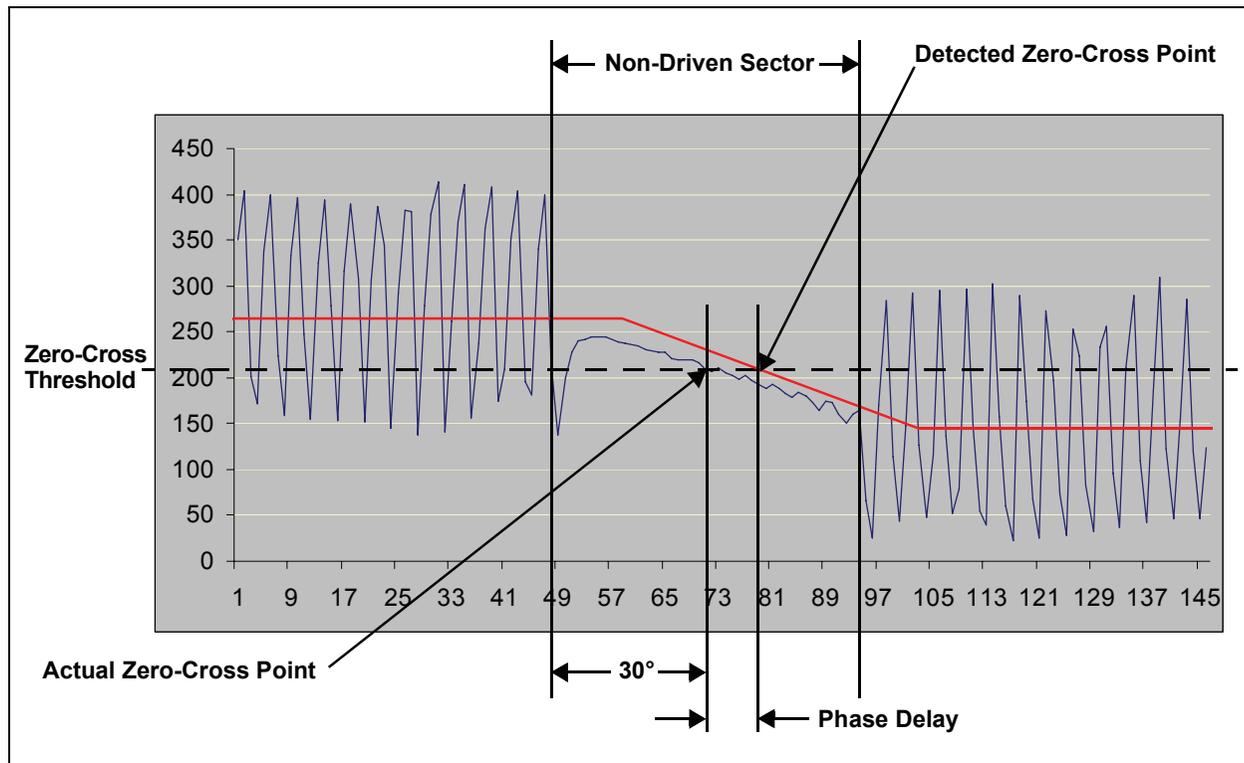
motor is commutated blindly until the back-EMF signal starts exhibiting the trapezoidal shape, shown in Figure 8. The back-EMF signal for a motor turning at low speed is shown in Figure 13.

At low speeds, there is a high probability of detecting the zero-cross event at some point other than half-way through the sector due to the shallow slope of the back-EMF signal. Any noise on the back-EMF signal, even the filtered signal, can result in premature zero-cross event detection.

It is important for the sensorless algorithm to sample and analyze all three back-EMF signals. By looking for zero-cross events on all three phases of the motor, the algorithm can recover in the next sector from a premature commutation initiated by early detection of a zero-cross event.

Figure 13 shows the phase delay of the ideal filtered signal. At low speed, the phase delay is small in comparison to the sector length. Note that a zero-cross event on the filtered back-EMF signal occurs well before the next commutation should occur. This is important, because at low speed, once the algorithm determines a zero-cross event has occurred, the next commutation can be planned for.

FIGURE 13: BACK-EMF AT LOW MOTOR SPEED



AN1083

High-Speed back-EMF Characteristics

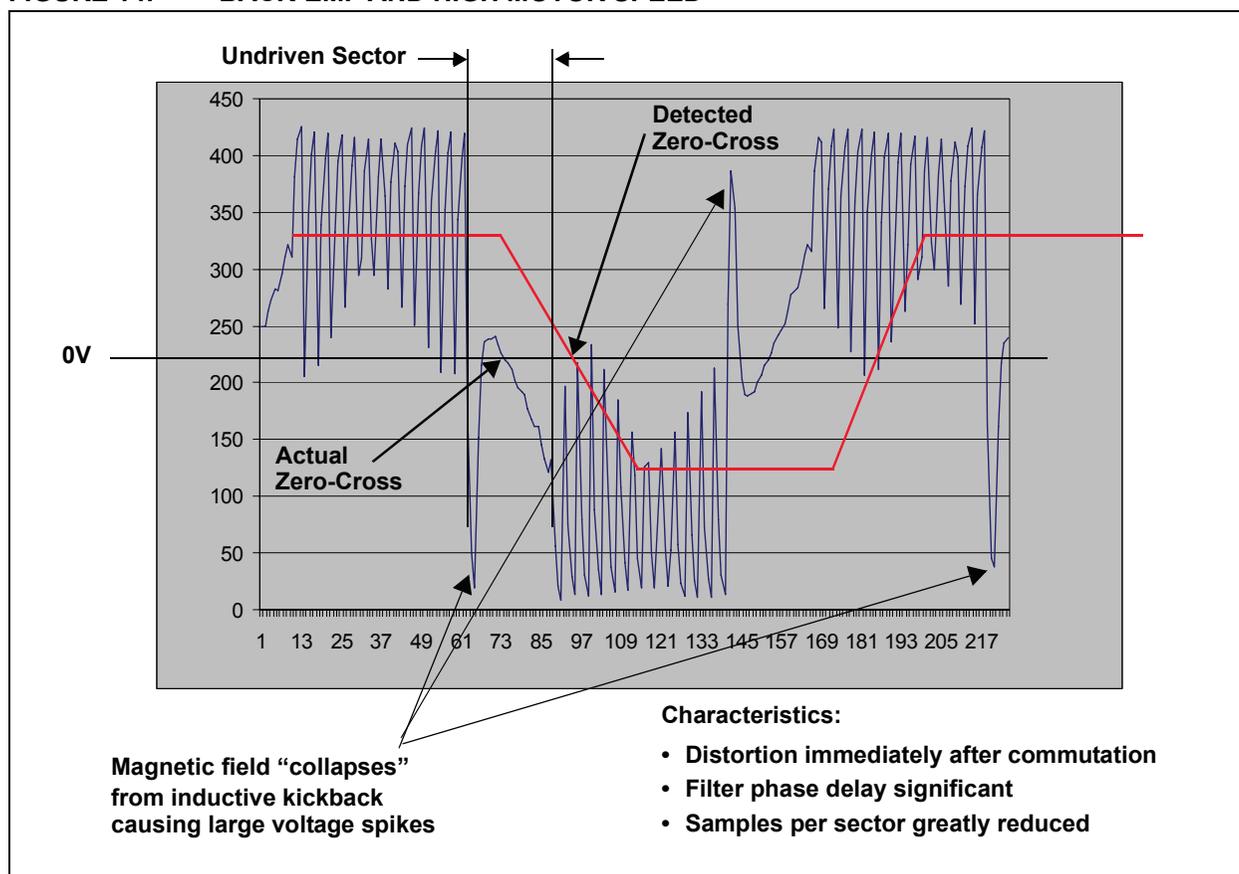
Back-EMF has different characteristics when the motor is turning fast, as shown in Figure 14. One obvious difference is the large voltage spike in the back-EMF signal at the start of the sector. This voltage spike is caused by inductive kickback. At higher speeds, more energy is stored in the motor windings. When a phase of the motor transitions from being driven to being tri-stated, the energy in the winding has to go somewhere; hence, the large voltage spike.

At high speed, the back-EMF signal slope is much steeper. As a result, it is much easier to detect a zero-cross event.

As motor speed increases, the time spent in each sector becomes shorter. This has two effects:

- Because the electrical revolution frequency is higher, the phase delay of the filter is much longer in comparison to sector width. Consequently, detection of the zero-cross event on the filtered signal occurs after the next commutation should transpire.
- Given a constant sample frequency over the entire speed range of the motor, fewer ADC samples of the BEMF are taken per sector at higher motor speeds. This means that, in relation to sector length, the resolution for detecting a zero-cross event decreases with motor speed.

FIGURE 14: BACK-EMF AND HIGH MOTOR SPEED



SENSORLESS IMPLEMENTATION

Based on the characteristics of back-EMF at low and high motor speeds, two implementations were created to optimize both low-speed and high-speed performance of the sensorless algorithm. Transition from the low-speed to the high-speed implementation (and vice versa) is seamless and has built-in hysteresis. As motor speed increases, the transition from the low-speed to the high-speed implementation occurs at 300 electrical revolutions per second by default. As motor speed decreases, the transition from the high-speed to the low-speed implementation occurs at 200 electrical revolutions per second by default. The transition speed is configurable by the user (see the “**Tuning Procedure**” section for more information).

Low-Speed Implementation

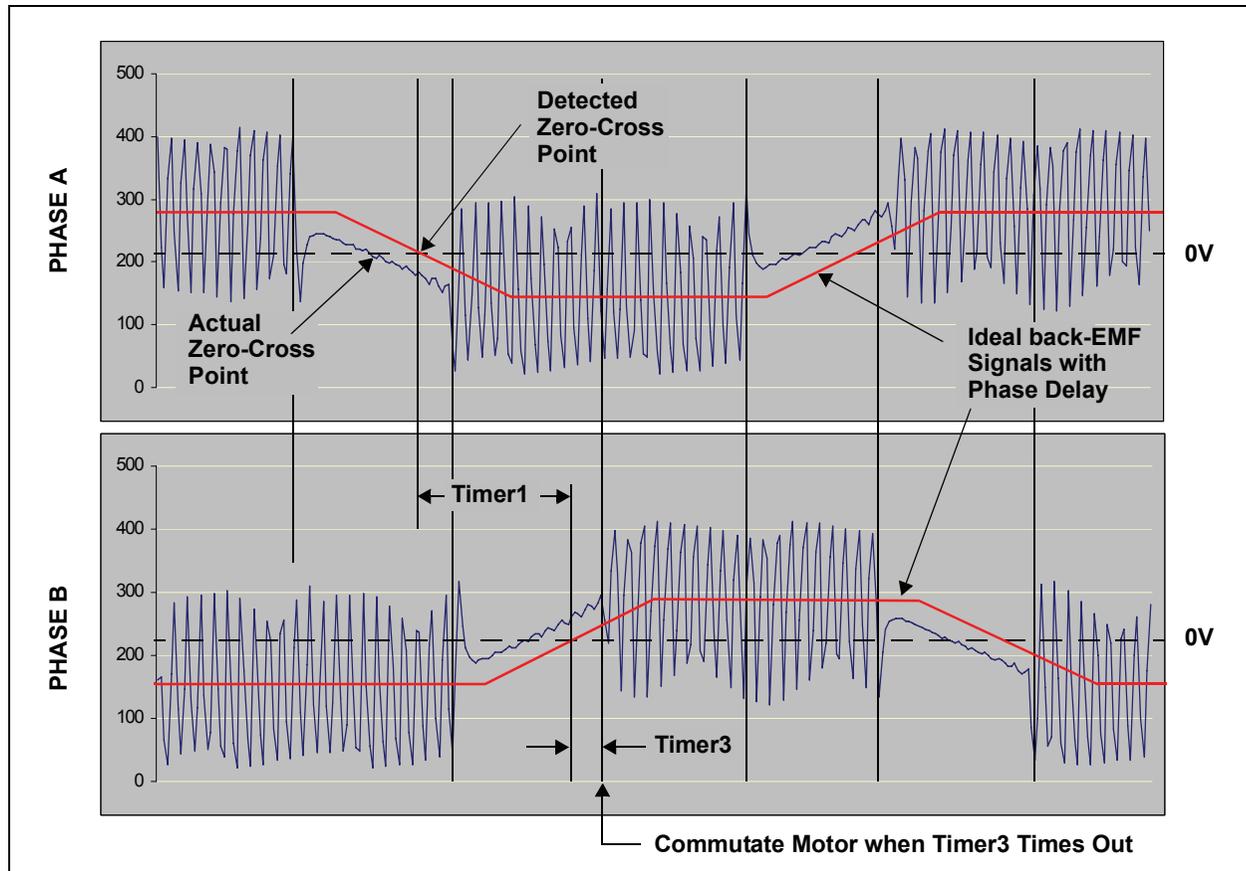
In the low-speed implementation of the algorithm, all three motor phases are sampled by the dsPIC30F ADC module. After the three samples are obtained, an ADC interrupt is generated. The samples are then fed through three identical IIR filters, one for each phase, to generate three filtered samples. Sampling occurs at a frequency of 49,152 Hz for each phase, which is more than two times faster than the PWM frequency (20 kHz).

Sampling is done during all sectors, driven and non-driven, for each phase. With sampling performed at greater than two times the PWM frequency, the filtered signal reflects the average potential applied to the motor winding during sectors when the phase is energized by the PWM signal. This helps create a filtered signal that resembles a trapezoidal shape.

Note: Technically, back-EMF refers only to the potential at the motor leads that results from the motor acting as a generator. In other words, when the PWM signal is applied to the motor – this is not back-EMF. For the sake of simplicity, however, this document describes continuous sampling of each phase as “sampling the back-EMF signal”.

Figure 15 shows the sampled back-EMF signal from two phases. The ideal filtered signal is transposed onto each (with phase delay). Zero-cross events are determined by comparing the filtered signal against the zero-cross threshold. The phase that the algorithm is currently analyzing, and whether the signal is rising or falling, depend on which sector the motor is currently in.

FIGURE 15: LOW-SPEED IMPLEMENTATION



AN1083

The dsPIC30F digital signal controllers have five onboard, 16-bit timers. Timer1 is used to measure the amount of time elapsed from one zero-cross event to the next. This time is equivalent to 60 electrical degrees. Assuming there is no phase delay when a zero-cross event is detected, the next commutation should occur in 30 degrees. Dividing the Timer1 capture value by two gives the time for 30 electrical degrees. Theoretically, this value can be loaded into the Timer1 Period register for another timer, referred to as the commutation timer. When the interrupt for the commutation timer occurs, it is time to commutate the motor windings to the next state.

Several sources of delay must be subtracted from the 30-degree time. The first is the phase delay of the digital filter. Using the dsPIC DSC filter design tool, this delay was approximated as 90 μ s.

Another source of delay is the time it takes to process the ADC interrupt. The ADC Interrupt Service Routine (ISR) executes three IIR filters before determining whether a zero-cross event has occurred. This process takes approximately 1.7 μ s. Each of these delays must be subtracted from the 30° time before it is loaded in the Commutation Timer Period register.

Timer3 is used as the commutation timer. When a zero-cross event occurs, the Timer3 Period register is loaded with the value specified by Equation 2.

EQUATION 2: TIMER3 PERIOD FOR LOW SPEED

$$PR3 = T_{30} - DFILT - DPROC - DPA$$

Where:

PR3 = Timer3 Period register value

T_{30} = Value computed for 30 electrical degrees

DFILT = Low-speed filter phase delay

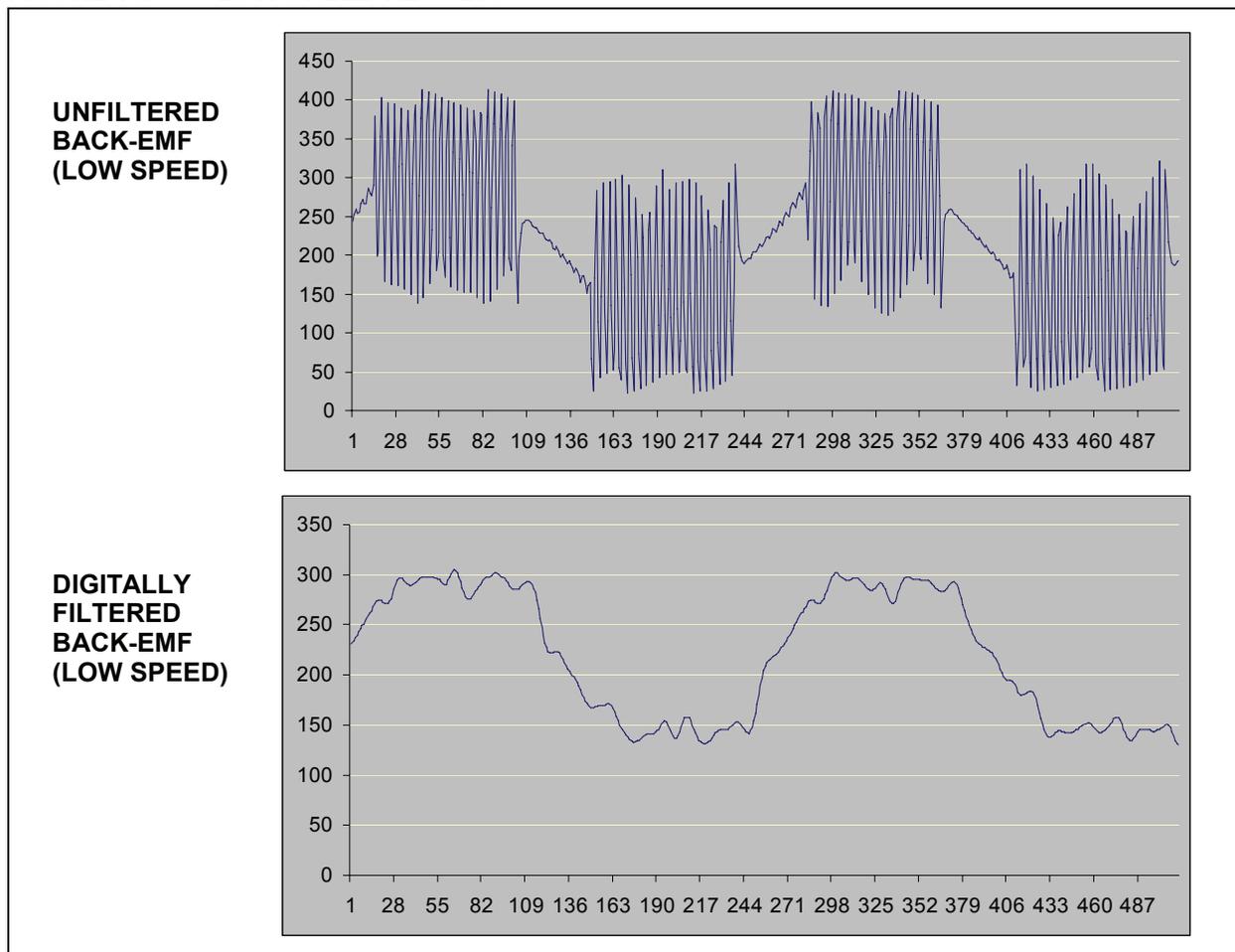
DPROC = Low-speed ADC interrupt processing delay

DPA = Phase advance (see the “Phase Advance” section for more information)

Timer3 and the Timer3 interrupt are then enabled. The motor windings are commutated to the next state when the Timer3 interrupt occurs.

The result of the low-speed implementation is shown in Figure 16.

FIGURE 16: LOW-SPEED RESULTS

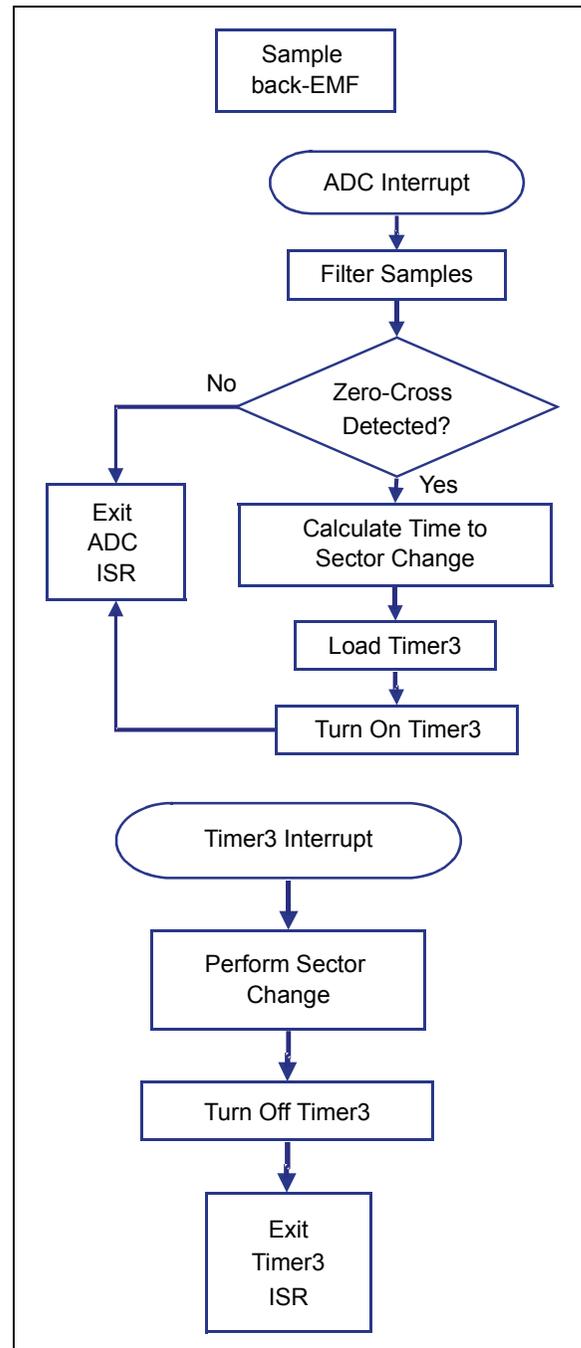


In summary, the low-speed implementation works as follows:

1. All three motor phases are sampled by the ADC module at a sample rate of 49,152 sps.
2. The samples for all three phases are filtered.
3. Based on the current motor state, the filtered sample from the appropriate phase is analyzed to see if a zero-cross event occurred.
4. When a zero-cross event is detected, the value in Timer1 is saved off and Timer1 is reset.
5. The saved Timer1 value, which is equivalent to the time for 60 electrical degrees, is divided by 2, and the IIR filter phase delay, processing delay and phase advance are subtracted. This result represents the amount of time until the next commutation should occur.
6. The result is loaded into the Timer3 Period register, Timer3 is turned on and the Timer3 interrupt is enabled.
7. When the Timer3 interrupt occurs, Timer3 is turned off and reset and the Timer3 interrupt disabled. The motor windings are commutated to the next state.

Figure 17 is a flowchart that illustrates the low-speed implementation of the sensorless BLDC algorithm.

FIGURE 17: FLOWCHART OF LOW-SPEED IMPLEMENTATION



High-Speed Implementation

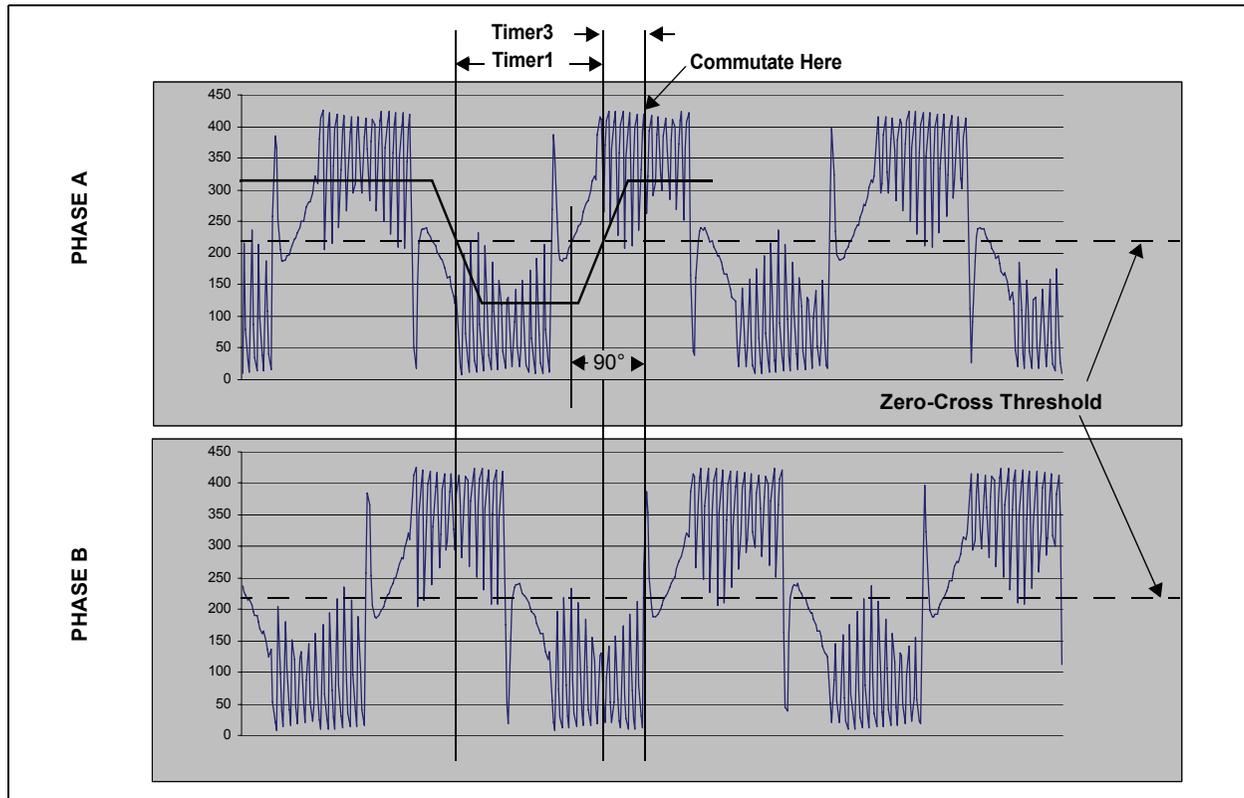
The high-speed implementation of this algorithm is shown in Figure 18. In this implementation, only one phase is sampled and filtered. The sampling frequency is increased to 81,940 sps.

The low-speed implementation will not spin the motor faster than about 40,000 electrical RPM (assuming no phase advance). At higher speeds, the filter and processing delays become larger than 30 electrical degrees (in time). Running faster causes zero-cross event detection to occur after the motor windings should have been commutated.

In the low-speed implementation, all three phases of the motor are sampled due to the shallow slope of the back-EMF signal at low motor speeds. A shallow slope in the back-EMF signal means a higher likelihood of detecting the zero-cross event prematurely. Looking for every zero-cross event minimizes the likelihood of the sensorless algorithm getting lost.

At high motor speeds, zero-cross event detection is much more accurate due to the steeper back-EMF signal slope. Therefore, sampling all three phases is not necessary. In addition, sampling only one motor phase frees up bandwidth to sample the one phase faster. This increases the accuracy of the zero-cross event detection.

FIGURE 18: HIGH-SPEED IMPLEMENTATION



In Figure 18, the zero-cross point on the filtered signal occurs more than 30 degrees after the actual zero-cross event. Rather than commutate the motor windings in the sector immediately following the actual zero-cross event, the high-speed implementation commutates the next sector. In other words, the algorithm looks forward 90 electrical degrees instead of 30.

Timer1 measures the time elapsed between zero-cross events on just one phase. This time is equivalent to 180 electrical degrees. Timer3 is then set up just as it was for the low-speed implementation, except the Timer3 Period register is loaded with the time for 90 electrical degrees minus the applicable delays. Equation 3 shows this relation.

EQUATION 3: TIMER3 PERIOD FOR HIGH SPEED

$$PR3 = T_{90} - DFILT - DPROC - DPA$$

Where:

- PR3 = Timer3 Period register value
- T_{90} = Value computed for 90 electrical degrees
- DFILT = High-speed filter phase delay
- DPROC = High-speed processing delay
- DPA = Phase advance delay (see the **“Phase Advance”** section for more information)

When the Timer3 interrupt occurs, the motor windings are commutated to the appropriate sector. The Timer3 interrupt is only commutating the motor every third sector. The other sectors are commutated by Timer2. After each zero-cross event detection, the Timer2 Period register is loaded with a value equivalent to the time for 60 degrees. Timer2 runs in the background, incrementally, commutating the motor windings to the next sector.

If Timer2 is free-running in the background, how is it referenced to an actual zero-cross event? The answer is that the Timer3 interrupt always forces a commutation every third sector to a particular sector state. When the Timer3 interrupt event occurs, Timer3 is zeroed and

turned off. In addition, Timer2 is also reset during a Timer3 interrupt. Therefore, Timer3 synchronizes Timer2 to the sector borders. Figure 19 illustrates how this works.

The inductive kickback seen in the back-EMF signals of the motor when it is turning at high speed was discussed earlier. This disturbance in the back-EMF signal is too large to filter out. It has to be "removed" from the pre-filtered signal or it will cause faulty zero-cross detection. To remedy this problem, the sensorless algorithm has a configurable parameter called the blanking count. The blanking count is the number of ADC samples that the algorithm should ignore at the start of each sector (see Figure 20).

FIGURE 19: THE JOB OF THE TIMERS IN HIGH-SPEED MODE

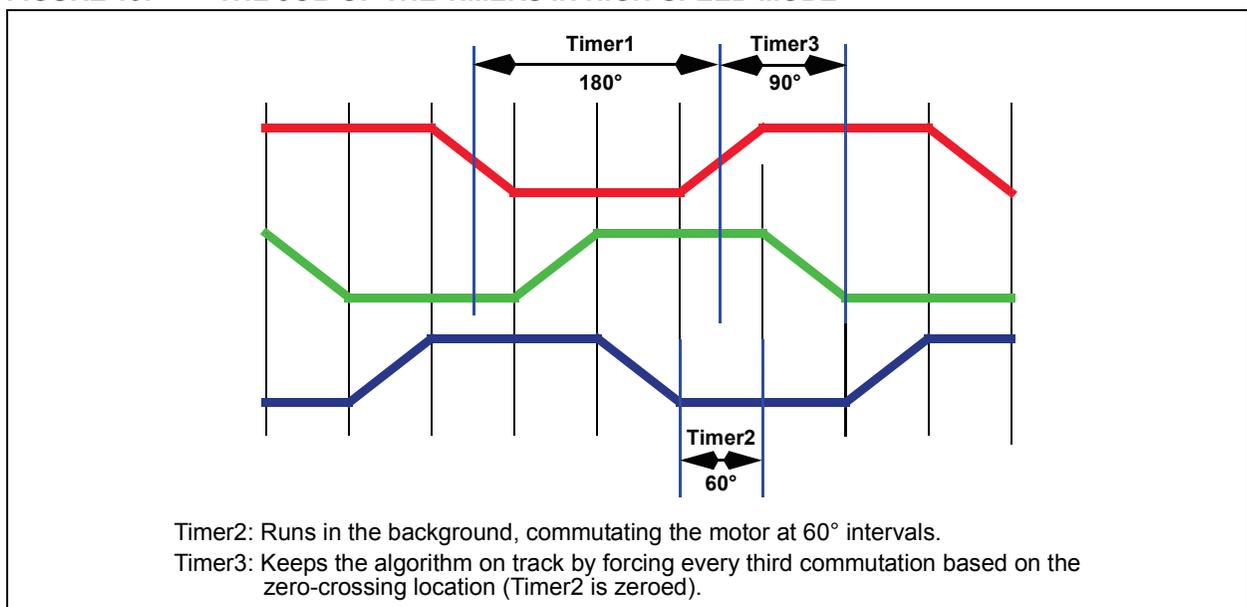
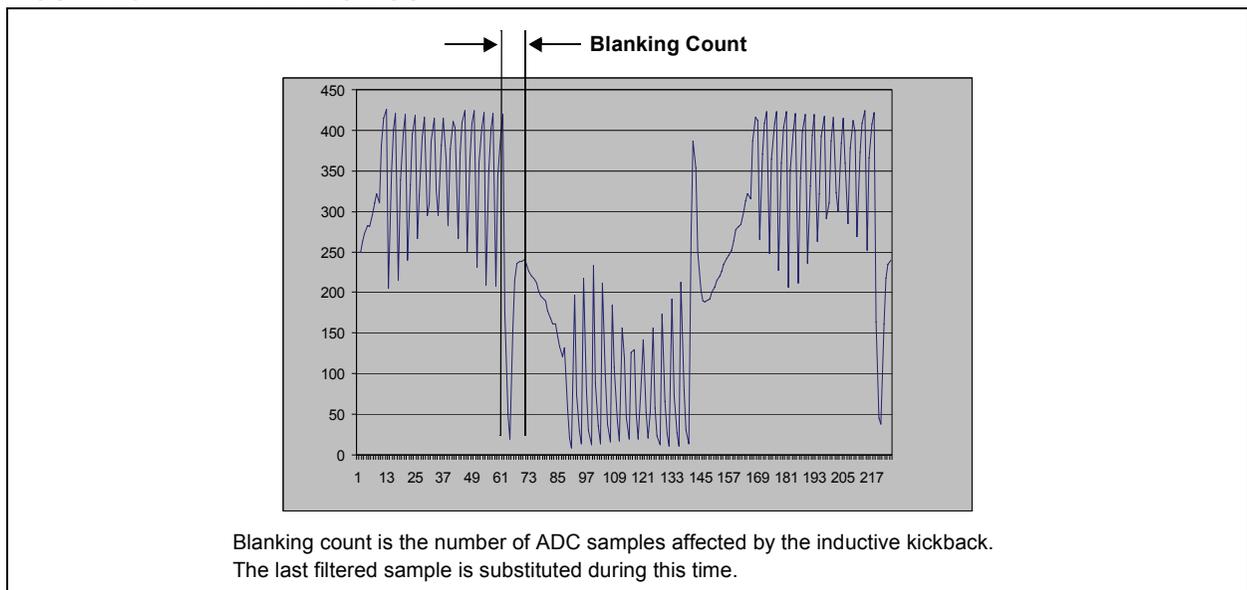


FIGURE 20: BLANKING COUNT

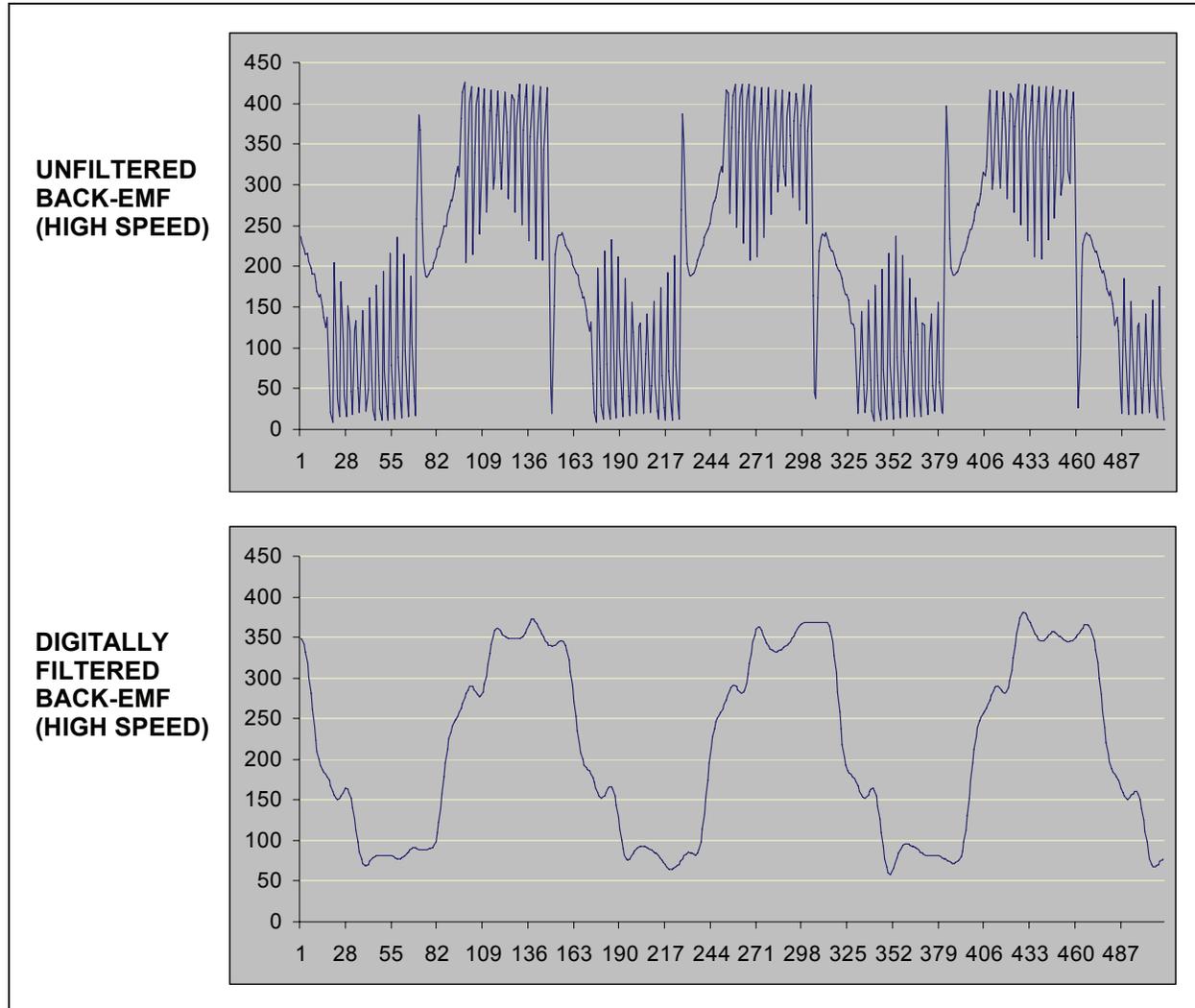


AN1083

In actuality, the algorithm does not ignore the back-EMF signal during the blanking period. Instead, at the start of each sector, the digital filter is fed the last filtered sample rather than the actual back-EMF sample for the duration of the blanking count.

The results of the high-speed implementation are shown in Figure 21.

FIGURE 21: HIGH-SPEED IMPLEMENTATION RESULTS



In summary, the high-speed algorithm works as follows:

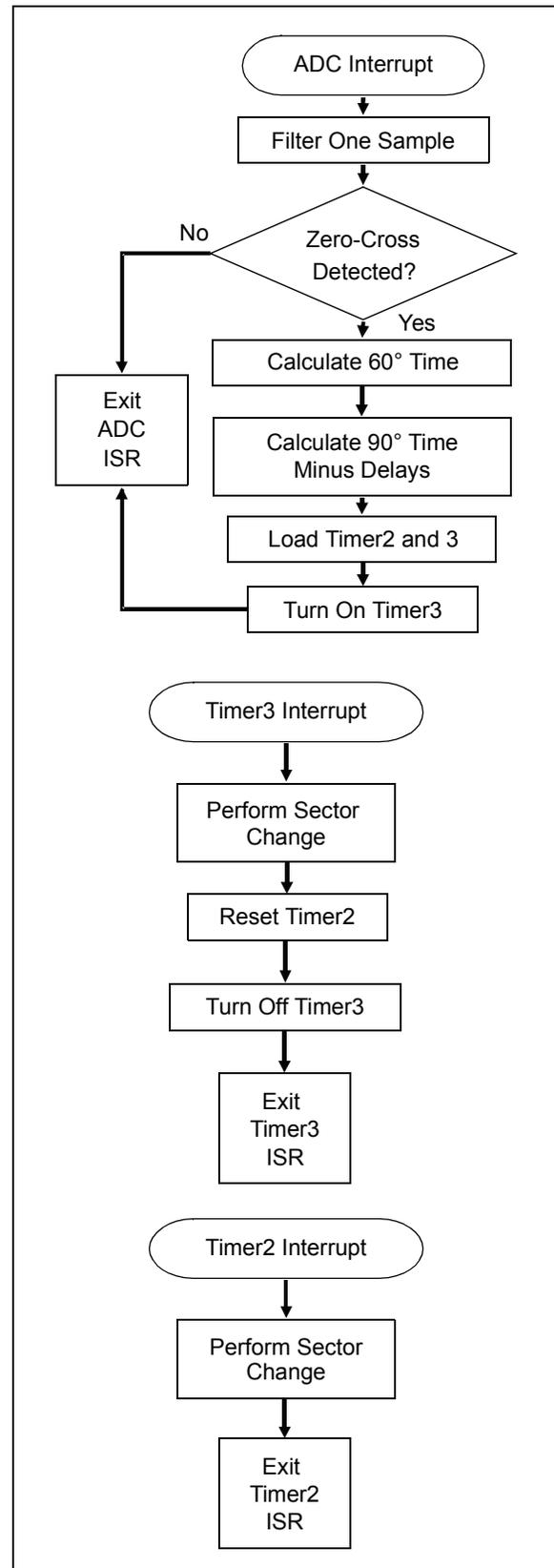
1. One motor phase is sampled using the ADC module at a sample rate of 81,940 sps.
2. The samples are filtered.

Note: The digital filter used in the high-speed implementation is the same as the filter described in the “**Digital Filtering**” section with one exception – the sampling frequency is 81,940 Hz instead of 49,152 Hz.

3. When a zero-cross event is detected, the value in Timer1 is saved and Timer1 is reset.
4. The Timer1 value, which is equivalent to the time for 180 electrical degrees, is divided by two, and the IIR filter phase delay, processing delay and phase advance delays are subtracted. This result represents the amount of time until the next commutation should occur (90 degrees after the actual zero-cross occurred).
5. The result is loaded into the Timer3 Period register, Timer3 is turned on and the Timer3 interrupt is enabled.
6. The Timer2 Period register is loaded with the time for 60 electrical degrees.
7. Timer2 runs continuously in the background. When a Timer2 interrupt occurs, the motor windings are commutated for the next sector.
8. When the Timer3 interrupt occurs, Timer3 is turned off and reset and the Timer3 interrupt disabled. The motor windings are commutated for a predetermined sector.
9. Timer2 is also reset. This ensures Timer2 is brought back in sync with the actual zero-cross event.

Figure 22 is a flowchart of the high-speed implementation of the sensorless BLDC algorithm.

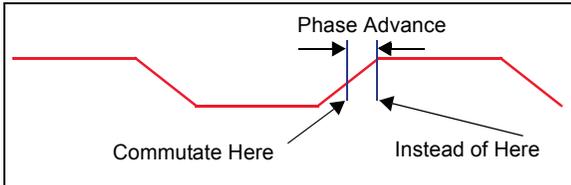
FIGURE 22: FLOWCHART OF HIGH-SPEED IMPLEMENTATION



Phase Advance

Phase advance is used to turn a motor faster than its rated speed. When phase advance is implemented, the motor windings are commutated early every sector. Phase advance is quantified in electrical degrees. Up to 30 degrees of phase advance is possible in this algorithm. Figure 23 illustrates phase advance.

FIGURE 23: PHASE ADVANCE



The increased motor speed that can be achieved when phase advance is implemented does come at a cost. The motor draws more current than it is rated for, which can cause overheating, thereby shortening the life of the motor.

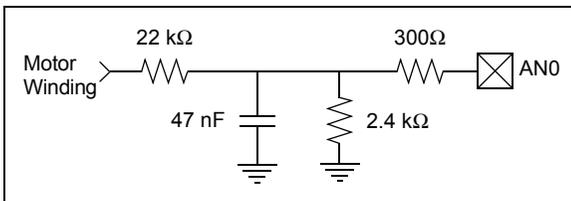
Note: Refer to Equation 2 and Equation 3 in the previous section to see how phase advance factors into the low-speed and high-speed implementations, respectively.

Zero-Cross Voltage Compensation

Ideally, the zero-cross voltage threshold is centered directly between the voltage rails driving the BLDC motor. For a motor being driven by $\pm 12\text{V}$, the zero-cross threshold would be 0V. For a motor being driven by a voltage range of 0-12V, the zero-cross voltage would ideally be 6V.

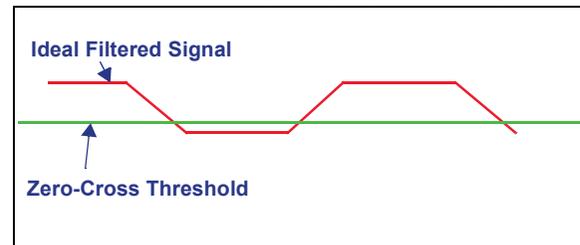
Figure 24 shows the circuit used to scale the back-EMF voltage before it is fed to an ADC channel on the dsPIC DSC. The voltage is scaled so that it is within the analog input voltage range of the dsPIC DSC.

FIGURE 24: BACK-EMF FEEDBACK CIRCUIT



Under ideal conditions, an ADC channel will sample the bus voltage and this result is divided by two (with positive and zero voltage drive rails) to determine the zero-cross threshold. This threshold is what the back-EMF filtered samples are compared against to determine whether a zero-cross event has occurred. If the filtered signal is not symmetrical about the zero-cross threshold, then zero-cross events cannot be determined accurately. Figure 25 shows what would happen if this were the case.

FIGURE 25: FILTERED BACK-EMF SIGNAL NOT SYMMETRICAL ABOUT ZERO-CROSS THRESHOLD



There are two filter sources for the back-EMF signal:

- The RC filter in the feedback circuit (see Figure 24) for filtering out high-frequency transients
- The digital low-pass filter

With low-pass filtering, the signal is not always symmetrical around the zero-cross voltage. As the motor speed increases, the back-EMF signal moves up slightly with respect to the zero-cross voltage.

To compensate, the back-EMF filtered samples are compared to the zero-cross threshold. A software accumulator tallies the number of samples larger than the zero-cross threshold vs. the number of samples smaller than the zero-cross threshold. If there is an imbalance, the accumulator will become a very large signed number over time. The higher order bits of the accumulator are added as an offset to the zero-cross threshold. This ensures that the back-EMF signal is always symmetrical about the zero-cross threshold.

Note: In the source code, the zero-cross threshold is stored in the `signal_average` variable. The calculation for `signal_average` is performed in the `CheckZeroCrossing()` function (`motor_isr.c`).

Speed Control

You have the option of enabling a PID speed control loop. The speed control loop is enabled via the Data Monitor and Control Interface tool. The “**Tuning Procedure**” section provides details on enabling and tuning the PID speed control loop.

Note: The PID function used in this algorithm is included as part of the DSP library provided with Microchip’s C30 C compiler. A detailed description of the PID function used can be found in the “*16-Bit Language Tools Libraries*” (DS51456).

The PID Speed control loop is called from the `MediumEvent()` function. `MediumEvent()` is called every 1 ms from the main program loop.

IMPLEMENTING THE ALGORITHM

To run the algorithm described in this application note, you must:

- Set up the hardware.
- Set up the development tools and load the project workspace in MPLAB IDE.
- Tune the motor using the Data Monitor and Control Interface (DMCI) tool.

Hardware Setup

The sensorless BLDC algorithm was developed for two hardware platforms:

- PICDEM™ Low-Voltage Motor Control (MC LV) Development Board (DM183021)
- dsPICDEM™ MC1 Motor Control Development Board (DM300020) in conjunction with the dsPICDEM™ MC1L 3-Phase Low-Voltage Power Module (DM300022).

These development boards, along with appropriate documentation, are available from the Microchip web site (www.microchip.com).

PICDEM™ MC LV DEVELOPMENT BOARD

This development board provides a motor terminal strip, 3-phase voltage source inverter bridge, motion sensor inputs, overcurrent protection, temperature sensor, push button switches, nine LEDs, test points for motor current and back-EMF sensing, a speed control potentiometer and an RS-232 port. The board must be configured as shown in Table 2. The motor should be connected as shown in Table 3.

TABLE 2: JUMPER SETTINGS FOR PICDEM™ MC LV DEVELOPMENT BOARD

| Jumper | Setting |
|--------|---------------|
| J7 | Short 2 and 3 |
| J8 | Open |
| J10 | Short |
| J11 | Short 2 and 3 |
| J12 | Open |
| J13 | Short 2 and 3 |
| J14 | Open |
| J16 | Short |
| J17 | Short |
| J19 | Open |
| J15 | Open |

TABLE 3: MOTOR CONNECTIONS TO PICDEM™ MC LV DEVELOPMENT BOARD

| Label on J9 | Connection |
|-------------|----------------------------|
| M1 | Phase A |
| M2 | Phase B |
| M3 | Phase C |
| G | Ground Wire (if available) |
| +(1) | 12-36V |
| -(1) | Ground |

Note 1: Connect board/motor power and ground to J9 only if the J20 connector is not used. Refer to the “*PICDEM™ MC LV Development Board User’s Guide*” (DS51554) for detailed information.

This sensorless BLDC motor control algorithm was developed to run at 30 MIPS. In order to achieve 30 MIPS, the 5 MHz crystal on the PICDEM MC LV board must be replaced with a 7.37 MHz crystal. This allows the algorithm to run at 29.49 MIPS.

Note: Limited testing has shown that the algorithm will run using the default 5 MHz crystal (20 MIPS). Running the code using the 5 MHz crystal requires that the user change the definition of `FCY` in `general.h` from 29490000 to 20000000.

Configuring the board as described in Table 2 and Table 3 connects the dsPIC30F3010 device so that the pins on the device function as shown in Table 4. This table shows only the pins that are used with this algorithm.

TABLE 4: dsPIC30F3010 PIN FUNCTIONS

| Pin | Name | Function |
|-----|-------|---|
| 1 | MCLR | Device reset via S1 |
| 2 | AN0 | Bus voltage sense |
| 3 | AN1 | Motor current sense |
| 4 | AN2 | Potentiometer (R14) input |
| 5 | AN3 | Phase 1 back-EMF sense |
| 6 | AN4 | Phase 2 back-EMF sense |
| 7 | AN5 | Phase 3 back-EMF sense |
| 8 | Vss | Ground |
| 9 | OSC1 | Crystal oscillator input |
| 10 | OSC2 | Crystal oscillator input |
| 11 | RC13 | Switch (S3) input |
| 12 | RC14 | Switch (S2) input |
| 13 | VDD | +5V |
| 14 | EMUD2 | In-circuit debugging data pin |
| 15 | EMUC2 | In-circuit debugging clock pin |
| 16 | FLTA | Overcurrent Fault input (current trip level set by R60) |
| 17 | PGD | In-Circuit Serial Programming™ data pin |
| 18 | PGC | In-Circuit Serial Programming clock pin |
| 19 | Vss | Ground |
| 20 | VDD | +5V |
| 21 | PWM3H | Drives high-side MOSFET (Q5) on Phase 3 |
| 22 | PWM3L | Drives low-side MOSFET (Q6) on Phase 3 |
| 23 | PWM2H | Drives high-side MOSFET (Q3) on Phase 2 |
| 24 | PWM2L | Drives low-side MOSFET (Q4) on Phase 2 |
| 25 | PWM1H | Drives high-side MOSFET (Q1) on Phase 1 |
| 26 | PWM1L | Drives low-side MOSFET (Q2) on Phase 1 |
| 27 | AVss | Ground |
| 28 | AVDD | +5V |

The “PICDEM™ MC LV Development Board User’s Guide” (DS51554) provides details on how to use the board for motors that require more than 36V.

Running the BLDC motor control algorithm on motors rated below 24V requires modifications to the board. Specifically, the resistor feedback network must be modified. The user’s guide also provides details about this modification.

The tactile switches on the PICDEM MC LV board provide the functionality listed in Table 5.

TABLE 5: SWITCH FUNCTIONALITY

| Button | Function |
|--------|---------------------|
| S1 | Device Reset |
| S2 | Begin start-up ramp |
| S3 | All stop |

dsPICDEM™ MC1 MOTOR CONTROL DEVELOPMENT BOARD AND THE dsPICDEM™ MC1L 3-PHASE LOW-VOLTAGE POWER MODULE

The dsPICDEM MC1 Motor Control Development Board includes the dsPIC30F6010A, various peripheral interfaces and a custom interface header system that allows different motor power modules to be connected to the PCB. The board also has connectors for mechanical position sensors, such as incremental rotary encoders and Hall effect sensors, and a breadboard area for custom circuits. The main control board receives its power from a standard plug-in transformer.

The dsPICDEM MC1L 3-Phase Low-Voltage Power Module is optimized for 3-phase motor applications that require a DC bus voltage less than 50 volts and can deliver up to 400W power output.

dsPICDEM™ MC1 Motor Control Development Board

The dsPICDEM MC1 Motor Control Development Board must be jumpered as shown in Table 6.

TABLE 6: dsPICDEM™ MC1 JUMPER SETTINGS

| Jumper | Setting |
|--------|---------------|
| LK1 | Open |
| LK2 | Open |
| LK3 | Open |
| LK4 | Short 2 and 3 |
| LK5 | Short 2 and 3 |
| LK6 | Open |
| LK7 | Open |
| LK8 | Open |
| LK9 | Open |

In addition, configure the dsPICDEM MC1 Motor Control Development Board as follows:

1. Set switch S2 to the ICD position.
2. Short R37 and R40. The MPLAB ICD 2 can not be used in Debug mode with the dsPIC30F6010A if these resistors are left in the circuit.
3. Make sure the DSC is a dsPIC30F6010A.
4. To filter out high-frequency transients on the back-EMF feedback lines, add 4 nF capacitors between ground and the AN12, AN13 and AN14 connections in the breadboard area.

The version of the sensorless algorithm that runs on the dsPICDEM MC1 Motor Control Development Board allows you to run a motor in Sensored mode. If you want this functionality, connect the Hall effect sensors as described in Table 7.

TABLE 7: HALL EFFECT SENSOR CONNECTIONS (OPTIONAL)

| Label on J3 | Connection |
|-------------|----------------------------|
| +5 | Hall effect supply voltage |
| G | Hall effect sensor ground |
| A | Phase R position hall |
| B | Phase Y position hall |
| C | Phase B position hall |

When the dsPICDEM MC1 Motor Control Development Board is configured as described above, **and** the Low-Voltage Power Module is configured as described in the “dsPICDEM™ MC1L 3-Phase Low-Voltage Power Module” section, the pins on the dsPIC30F6010A function as described in Table 8.

TABLE 8: dsPIC30F6010A PIN FUNCTIONS

| Pin | Name | Function |
|-----|----------|--|
| 6 | RG6 | Switch (S4) input |
| 7 | RG7 | Switch (S5) input |
| 8 | RG8 | Switch (S6) input |
| 9 | MCLR | Device Reset via S1 |
| 10 | RG9 | Switch (S7) input |
| 11 | Vss | Ground |
| 12 | VDD | +5V |
| 13 | FLTA | Shunt overcurrent/overvoltage/ Hall (overcurrent)/brake chopper Fault input from LV power module |
| 14 | RE9 | Reset power module Faults |
| 19 | PGC/EMUC | In-Circuit Serial Programming™/ debugging clock pin |
| 20 | PGD/EMUD | In-Circuit Serial Programming/ debugging data pin |
| 22 | AN7 | Potentiometer (VR2) input |
| 23 | RA9 | LED (D6) control |
| 24 | RA10 | LED (D7) control |
| 25 | AVDD | +5V |
| 26 | AVSS | Ground |
| 27 | AN8 | Bus current sense |
| 30 | AN11 | Bus voltage sense |
| 31 | Vss | Ground |
| 32 | VDD | +5V |
| 33 | AN12 | Phase 1 (R) back-EMF sense |
| 34 | AN13 | Phase 2 (Y) back-EMF sense |
| 35 | AN14 | Phase 3 (B) back-EMF sense |
| 48 | VDD | +5V |
| 49 | OSC1 | Crystal oscillator input |
| 50 | OSC2 | Crystal oscillator input |
| 51 | Vss | Ground |
| 52 | RA14 | LED (D8) control |
| 53 | RA15 | LED (D9) control |
| 54 | RD8 | Phase 1 (R) position Hall effect sensor |

TABLE 8: dsPIC30F6010A PIN FUNCTIONS (CONTINUED)

| Pin | Name | Function |
|-----|-------|--|
| 55 | RD9 | Phase 2 (Y) position Hall effect sensor |
| 56 | RD10 | Phase 3 (B) position Hall effect sensor |
| 69 | RD7 | LED (D5) control |
| 70 | Vss | Ground |
| 71 | VDD | +5V |
| 76 | PWM1L | Drives low-side MOSFET (Q4) on Phase 1 (R) |
| 77 | PWM1H | Drives high-side MOSFET (Q3) on Phase 1 (R) |
| 78 | PWM2L | Drives low-side MOSFET (Q6) on Phase 2 (Y) |
| 79 | PWM2H | Drives high-side MOSFET (Q5) on Phase 2 (Y) |
| 80 | PWM3L | Drives low-side MOSFET (Q8) on Phase 3 (B) |
| 1 | PWM3H | Drives high-side MOSFET (Q7) on Phase 3 (B) |

The tactile switches on the dsPICDEM MC1 board have the following functionality:

TABLE 9: SWITCH FUNCTIONALITY

| Button | Function |
|--------|---------------------|
| S1 | Device Reset |
| S3 | Simulate Fault |
| S4 | Hall mode |
| S5 | Begin start-up ramp |
| S6 | Sensorless mode |
| S7 | All stop |

dsPICDEM™ MC1L 3-Phase Low-Voltage Power Module

The Low-Voltage Power Module must be configured for the sensorless algorithm, which requires that the metal lid on the power module be removed.

WARNING

Removing the top of the metal enclosure may result in electric shock. Wait at least 5 minutes after power is removed from the module before contacting the PCB in the enclosure. Read all warnings in the “dsPICDEM™ MC1L 3-Phase Low-Voltage Power Module User’s Guide” (DS70097) before removing the top of the metal enclosure.

After removing the lid on the Low-Voltage Power Module, solder 51 ohm resistors across the links shown in Table 10.

AN1083

TABLE 10: LINKS POPULATED WITH 51 OHM RESISTORS

| Link | Connection |
|------|-----------------------------|
| LK22 | Bus current sense |
| LK24 | Phase 3 (B) back-EMF sense |
| LK25 | Phase 2 (Y) back-EMF sense |
| LK26 | Phase 1 (R) back-EMF sense |
| LK30 | Bus voltage sense |

These are the only modifications required if the voltage supplied to the Low-Voltage Power Module is 24V to 48V.

Additional modifications are required if: (a) the supplied voltage is lower than 24V, or (b) a higher current level is required. For details on making these modifications, refer to the “*dsPICDEM™ MC1L 3-Phase Low-Voltage Power Module User’s Guide*” (DS70097).

After these modifications are made, secure the enclosure lid using the screws set aside during removal.

Table 11 lists the screw terminal connections. Make these connections only after the lid of the Low-Voltage Power Module is securely in place.

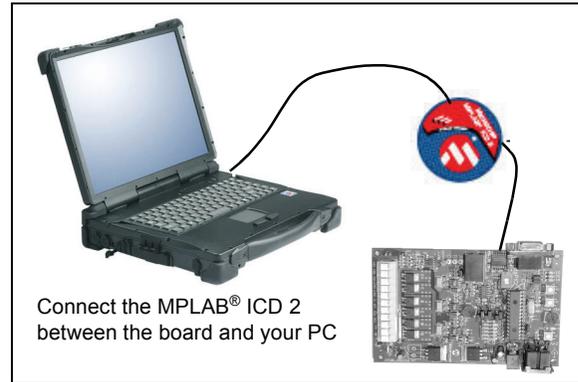
TABLE 11: SCREW TERMINAL CONNECTIONS

| Label | Screw Terminal Connection |
|------------|---------------------------|
| + | Positive Voltage Rail |
| - | Negative Voltage Rail |
| Earth | Earth Ground (optional) |
| R | Motor Phase 1 (R) |
| Y | Motor Phase 2 (Y) |
| B | Motor Phase 3 (B) |
| Gnd Symbol | Motor Ground |

Development Tool Setup

After connecting your PC to the development board via the MPLAB ICD 2 Debugger, follow the procedures below for your board to load the algorithm workspace and load the DMCI tool.

FIGURE 26: DEVELOPMENT TOOL SETUP



PICDEM™ MC LV Development Board

Step One – Configure the Workspace:

1. Unzip the AN1083 MCLV.zip file.
2. Open MPLAB IDE (version 7.50 or later).
3. Open the Sensorless.mcw workspace.
4. From the Tools menu, select the DMCI tool.
5. Load AN1083 MCLV.DMCI in the DMCI tool.

Step Two – Program the dsPIC30F Device:

1. Place S4 in the PRGM position.
2. Select the ICD 2 as the debug tool.
3. Connect to the ICD 2.
4. Select “Debug” from the Project Manager toolbar.
5. Build the code.
6. Program the device.

Step Three – Run the Algorithm:

1. Place S4 in the DEBUG position.
2. Reset the device.
3. Run the debugger.
4. Halt the debugger.

dsPICDEM™ MC1 MOTOR CONTROL DEVELOPMENT BOARD AND THE dsPICDEM™ MC1L 3-PHASE LOW-VOLTAGE POWER MODULE

Step One – Configure the Workspace:

1. Unzip the AN1083_MC1.zip.
2. Open MPLAB IDE (version 7.50 or later).
3. Open the `Sensorless.mcw` workspace.
4. From the Tools menu, select the DMCI tool.
5. Load AN1083_MC1.DMCI in the DMCI tool.

Step Two – Program the dsPIC30F Device:

1. Place S2 in the ICD position.
2. Select the ICD 2 as the debug tool.
3. Connect to the ICD 2.
4. Select “Debug” from the Project Manager toolbar.
5. Build the code.
6. Program the device.

Step Three – Run the Algorithm:

1. Reset the device.
2. Run the debugger.
3. Halt the debugger.

Data Monitor and Control Interface Tool

The Data Monitor and Control Interface (DMCI) tool eliminates hours of trial and error setup by providing a graphical user interface to configure the motor parameters. This tool is included in MPLAB IDE version 7.50 or later. The DMCI tool provides sliders and text box controls that let you assign global variables. You can easily change these variables by adjusting the sliders or changing text fields. The tool also lets you assign arrays to one of four graphs.

When you have optimized the motor parameters, the graphical interface settings (the variables assigned to different controls) can be saved so that other people have access to the setup.

The DMCI consists of these screens and controls:

- Dynamic Data Input Screen
- Dynamic Data Control Screen
- Dynamic Data View Screen

DYNAMIC DATA INPUT SCREEN

The Dynamic Data Input screen (Figure 27) is used to define the motor characteristics. Table 12 identifies the controls used for the sensorless algorithm.

DYNAMIC DATA CONTROL SCREEN

The Dynamic Data Control screen (Figure 28) is used to set up the sensorless algorithm parameters. Table 13 identifies the controls used.

FIGURE 27: DYNAMIC DATA INPUT SCREEN

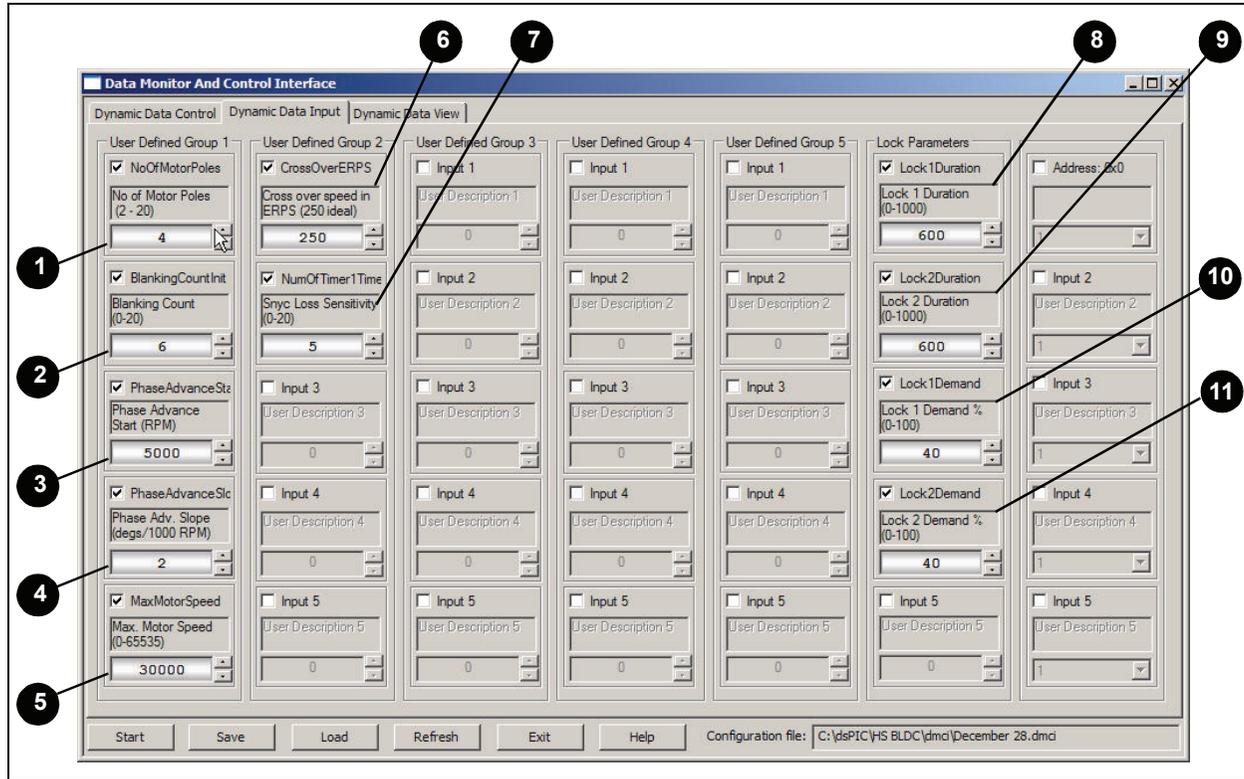


TABLE 12: DYNAMIC DATA INPUT SCREEN CONTROLS

| Reference | Control Function |
|-----------|---|
| 1 | Number of motor poles (should be an even number). |
| 2 | Blanking count (0-20) in 12 ms counts. ⁽¹⁾ |
| 3 | Motor speed (in RPM) at which phase advance should start. |
| 4 | Phase advance slope in degrees per 1000 RPM (up to 30 degrees). |
| 5 | Max. motor speed in RPM. This number is used to set the scale of the full range of the control potentiometer when the PID speed control loop is enabled. ⁽²⁾ |
| 6 | Speed (electrical RPS) at which the algorithm transitions from the low-to-high-speed implementation or vice versa. The default value is 250 and typically does not need to be changed. The crossover speed can not be less than 50 because there are ± 50 electrical revolutions per second of built-in hysteresis. |
| 7 | Shutdown sensitivity specified in the number of Timer1 overflows. If Timer1 overflows, the motor has stopped or the algorithm has lost sync. Increase this number to make the algorithm less sensitive. |
| 8 | The duration of Lock 1 specified in milliseconds. ⁽³⁾ |
| 9 | The duration of Lock 2 specified in milliseconds. ⁽³⁾ |
| 10 | The demand (PWM duty cycle in %) for Lock 1. ⁽³⁾ |
| 11 | The demand (PWM duty cycle in %) for Lock 2. ⁽³⁾ |

Note 1: 12 ms corresponds to the ADC interrupt frequency (approximately) when the algorithm is in High-Speed mode.

2: VR2 on the dsPICDEM™ MC1 Motor Control Development Board; R14 on the PICDEM™ MC LV Development Board.

3: The lock parameters are part of the open-loop start-up ramp. The purpose of the lock time periods is to drive the motor to Sector 0 (Lock 1), and then to Sector 1 (Lock 2) for a small amount of time, so that the motor will align to a known sector before the start-up ramp begins.

FIGURE 28: .FIGURE 21: DYNAMIC DATA CONTROL SCREEN SETUP

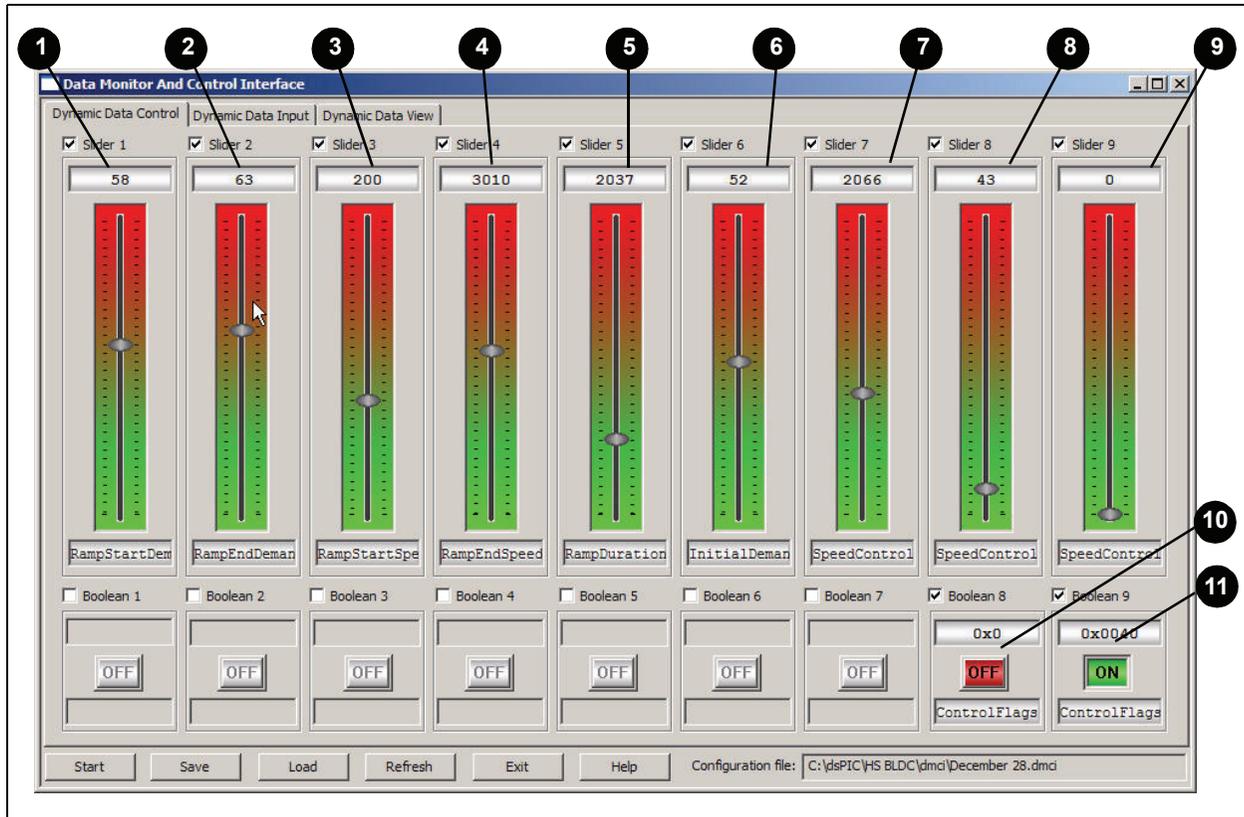


TABLE 13: DYNAMIC DATA CONTROL SCREEN CONTROLS

| Reference | Control Function |
|-----------|--|
| 1 | Ramp start demand (PWM duty cycle in % at the start of the open-loop start-up ramp). |
| 2 | Ramp end demand (PWM duty cycle in % at the end of the open-loop start-up ramp). |
| 3 | Ramp start speed in RPM. |
| 4 | Ramp end speed in RPM. |
| 5 | Ramp duration in ms (0.5 to 6.5 seconds). |
| 6 | Initial demand (sets the demand immediately after the start-up ramp). |
| 7 | Speed control loop P error gain term. |
| 8 | Speed control loop I error gain term. |
| 9 | Speed control loop D error gain term. |
| 10 | Speed control loop enable. When in the ON state, the control potentiometer is used to adjust motor speed. When in the OFF state, the control potentiometer adjusts PWM duty cycle. ⁽¹⁾ |
| 11 | Control potentiometer enable/disable. When in the ON state, the potentiometer adjusts the speed reference of the PID speed control loop or PWM duty cycle (refer to control #10 above). When in the OFF state, the control potentiometer has no effect on motor speed. |

Note 1: VR2 on the dsPICDEM™ MC1 Motor Control Development Board; R14 on the PICDEM™ MC LV Development Board.

AN1083

DYNAMIC DATA VIEW SCREEN

Note: This section is only applicable for sensorless control with the dsPIC30F6010A on the dsPICDEM MC1 platform.

The third tab on the DMCI tool accesses the Dynamic Data View screen (Figure 29). By default, this screen graphically shows four 512-word buffers. The graphs display the following information:

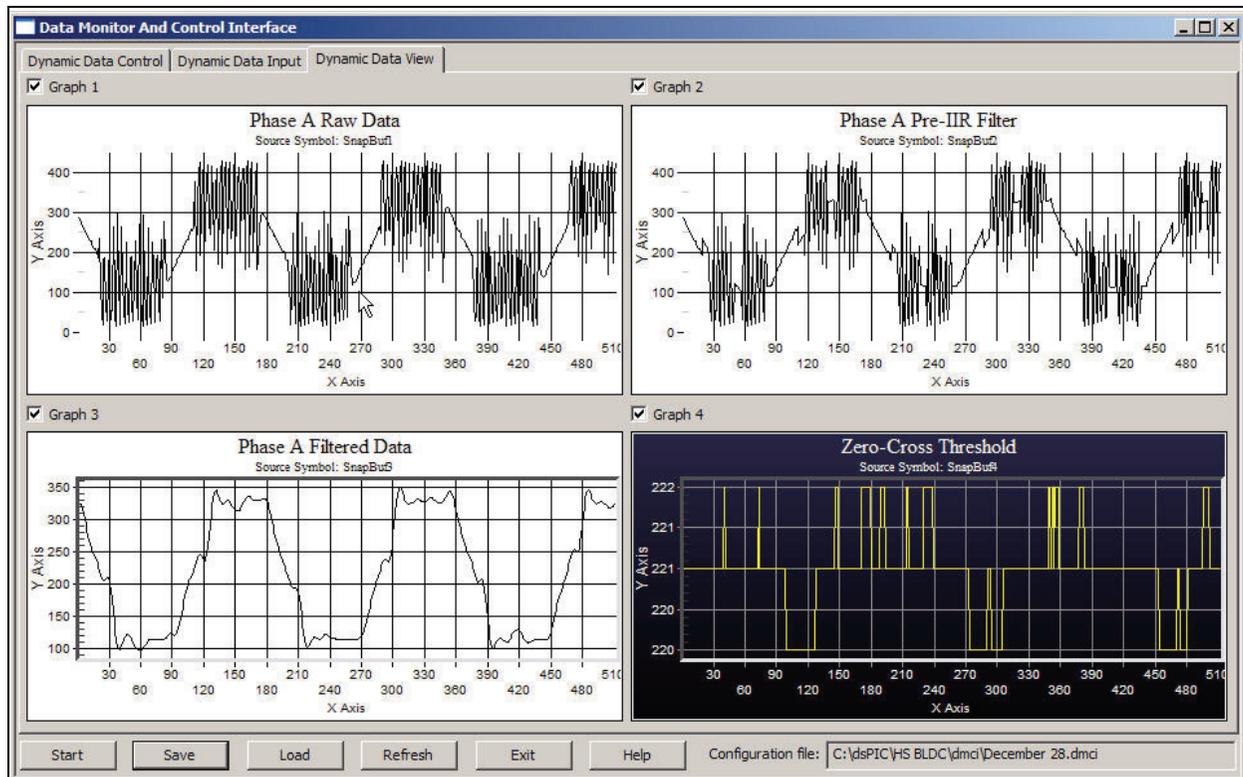
- Graph 1: Phase A Raw back-EMF Samples
- Graph 2: Phase A Raw back-EMF Samples (blanking count samples ignored)
- Graph 3: Phase A Filtered Samples
- Graph 4: Zero-Cross Reference

When the motor is running in Sensorless mode, pressing S6 on the board initiates a buffer capture. When the debugger is halted, the graphs display the buffered back-EMF signal (and the three other buffers).

The DMCI tool lets you zoom in on areas of each graph by dragging a box around the desired area.

You can enlarge a graph to full screen by selecting it and pressing the <M> key. Pressing <Esc> returns you to the DMCI GUI.

FIGURE 29: DYNAMIC DATA VIEW SCREEN



Tuning Procedure

To tune the motor parameters you must follow this precise process:

1. Be sure the motor is stopped (failure to do so will cause a current spike when the debugger is restarted).
2. Halt the MPLAB ICD 2 debugger.
3. Change the appropriate parameter.
4. Restart the MPLAB ICD 2 debugger.
5. Start the motor.

This process must be repeated any time you adjust a motor parameter. Table 14 details which switches start and stop the motor for each development board.

TABLE 14: MOTOR CONTROL SWITCHES

| Development Board Type | On | Off |
|------------------------|----|-----|
| dsPICDEM™ MC1 | S5 | S7 |
| PICDEM™ MC LV | S2 | S3 |

The method used to tune the start-up ramp and PID speed control parameters follows these basic steps:

1. Set up the initial parameters that match the characteristics of the motor.
2. Initialize the open-loop start-up ramp.
3. Tune the open-loop start-up parameters.
4. Tune the speed control loop.

Follow these detailed procedures:

Step One: Set up the initial parameters that match the characteristics of the motor.

1. On the Dynamic Data Input screen, enter the number of motor poles and maximum speed of the motor. Leave the blanking count at the default value.
2. Disable phase advance during the initial tuning of motor parameters by setting the phase advance start speed to an RPM value that is higher than the maximum motor speed.
3. Set the phase advance slope to zero. This ensures that no phase advance is added during the initial tuning process.
4. The crossover speed should be 250. If you have a lower speed motor (40k electrical RPM maximum) that does not require phase advance, you can change this to a very high number, such as 700, to make sure the algorithm runs in Low-Speed mode all the time.
5. Enter 20 for the shutdown sensitivity. This makes the algorithm less sensitive to shutdown detection (the lower this number the more sensitive the algorithm).

Step Two: Initialize the open-loop start-up ramp.

1. Set the Lock 1 and Lock 2 duration values at 600 ms.
2. Set the lock demands. It is best to start out at lower demand values. Start at a lock value of 10 for Lock 1 and Lock 2. Increase the Lock 1 and Lock 2 demands until the motor snaps to the lock positions, but it is not drawing sufficient current to damage the motor windings or the drive circuitry. The whole purpose of the lock positions is to bring the motor to a known sector before beginning the start-up ramp.
3. The remaining start-up ramp parameters are found on the Dynamic Data Control screen. In the lower right hand corner of this screen, disable the speed control loop and the control potentiometer by setting the enabled boolean buttons to "off". The idea is to get the sensorless algorithm to run the motor before the algorithm has to compensate for changes in demand.
4. Set the ramp duration to the full duration – 6500 ms. Once the start-up ramp is tuned so that motor start-up is successful, the ramp duration can be reduced.

Step Three: Tune the open-loop start-up parameters.

1. Ignore the PID gain sliders for the moment. Concentrate on getting the ramp start speed, end speed, start demand and end demand set so the that motor spins fast enough for the sensorless algorithm to sync up.
2. Use the sliders on the Dynamic Data Control screen to set the start and end speeds. Set the end speed to something on the order of 12-20% of full motor speed.

For motors with a lot of inertia and/or many motor poles, the start speed will have to be less than 100 RPM. For motors with little inertia and/or few motor poles, the start speed can be higher (typically between 200 and 300 RPM).

3. Keep the start, end and initial demand sliders the same. Set all three parameters to the same demand value specified for Lock 1 and Lock 2.

Click 'start' on the DMCI tool, then press the button on the board that initializes the start-up ramp (this is hardware platform dependant).

If the motor sits in place and shutters, more demand is needed. Press the button on the board to stop the motor. Click 'halt' on the DMCI tool. Increase the start, end and initial demand by units of 5 and repeat this step until the motor starts.

Note: You should increase demand slowly. If you jump directly to a large demand value, the motor will draw a lot of current which could harm the motor or drive circuitry.

- The motor should now start turning when the motor start button is pressed. If motor commutations appear to occur faster than the motor can keep up with, increase the end and initial demands.

The end demand should never be greater than 20% higher than the start demand. Typically, the initial demand should be slightly lower than the end demand. If the motor still appears to “slip”, decrease the end speed.

- The motor should now spin up, and the sensorless algorithm should take over at the end of the start-up ramp. Fine tune the ramp parameters until motor start-up is reliable.
- Experiment with decreasing the ramp duration. Find the minimum value of the ramp duration that will still allow the motor to start-up reliably.
- It is now time to enable the control potentiometer. Enable the control potentiometer on the DMCI tool.

Do not enable speed control. The potentiometer position will correlate to the PWM duty cycle; full clockwise is 100% duty cycle.

Estimate the potentiometer position that corresponds to the initial demand value. Set the potentiometer in this position and initiate the start-up ramp. If the motor doesn’t start, adjust the potentiometer and try again. When the motor does run, the speed is adjustable by turning the potentiometer.

Step Four: Tune speed control loop.

- The speed control loop can now be enabled. Enable the speed control loop on the DMCI interface. The potentiometer position now correlates to motor speed (in RPM). Full clockwise corresponds to the maximum motor speed specified on the Dynamic Data Input screen of the DMCI tool.
- Move the initial demand slider to about 80% of the end demand. Then, zero the integral and derivative terms and center the proportional slider. Estimate the potentiometer position that corresponds to the end speed of the start-up ramp. Try running the motor. Adjust the control potentiometer and the P term until the motor starts successfully. (Lowering the initial start demand can help improve start-up performance.) Some Integral gain may be needed to start the motor successfully.
- Tune the P, I and D terms until the desired motor performance is obtained. Very little integral term is needed in comparison to the proportional term.

See the “**Troubleshooting**” section for tips on possible problem solutions.

Changing the Digital Filter Coefficients

The low and high-speed filter coefficients are located in `BEMF_filter.s`. These filter coefficients can be replaced by new coefficients generated by the dsPIC DSC Filter Design tool.

Other Tunable Features of the Algorithm

Table 15 lists some of the configurable #define statements and where they are located in the source code.

Note: Altering these features is not recommended. Adjust these features at your own risk.

TABLE 15: OTHER CONFIGURABLE PARAMETERS.

| Parameter | Defined In | Description |
|--|-------------------------|--|
| FCY | <code>general.h</code> | The processor frequency in instructions per second. |
| FPWM | <code>general.h</code> | PWM frequency. |
| FULL_DUTY | <code>general.h</code> | The PWM period (a function of FCY and FPWM). |
| FILTER_PHASE_DELAY | <code>general.h</code> | Filter phase delay in Timer3 counts. |
| PROCESSING_DELAY | <code>general.h</code> | ADC interrupt processing delay in Timer3 counts for Low-Speed mode. |
| PROCESSING_DELAY_HS | <code>general.h</code> | ADC interrupt processing delay in Timer3 counts for High-Speed mode. |
| CYCLE_BY_CYCLE_PROTECTION FAULT_CAUSE_PWM_SHUTDOWN NO_FAULT_PROTECTION | <code>pwm.c</code> | Three choices for what the PWM module will do when a Fault condition occurs. Change the “FLTACON =” to one of these choices. |
| DEAD_TIME | <code>pwm.c</code> | PWM dead time in seconds. |
| SNAPSHOT | <code>snapshot.h</code> | When defined RAM is allocated for the Dynamic Data View graphs. |
| SNAPSIZE | <code>snapshot.h</code> | The size of the four snapshot buffers in bytes. |

CONCLUSION

This application note is intended for the developer who wants to drive a sensorless BLDC motor without the use of discrete, low-pass filtering hardware and off-chip comparators.

The algorithm described uses digital filtering of the back-EMF signals generated by a turning BLDC motor. Digital filtering makes it possible to more accurately detect the zero-cross events in the back-EMF signal. When detected by the dsPIC DSC, zero-cross events provide the information needed by the algorithm to commutate the motor windings

Accurately detecting the zero-cross events in a back-EMF signal is the key to sensorless control of a BLDC motor that is driven using six-step, or trapezoidal, commutation. The use of digital filtering, as opposed to hardware filters or external comparators, requires less hardware, which equates to less cost and a smaller PCB. The use of an open source solution equates to a shorter development cycle

Finally, the Microchip development tools that support the dsPIC DSC device family, and specifically, the Data Monitor and Control Interface tool included with MPLAB IDE version 7.50 or later, allow the algorithm to be quickly adapted to your specific motor and application.

RESOURCES

For BLDC motor control, see:

- AN901, “Using the dsPIC30F for Sensorless BLDC Control”
- AN957, “Sensored BLDC Motor Control Using dsPIC30F2010”
- AN1017, “Sinusoidal Control of PMSM Motors with dsPIC30F DSC”
- AN1078, “Sensorless Field Oriented Control of PMSM Motors”

For information on the hardware platforms compatible with this algorithm, see:

- “dsPICDEM™ MC1 Motor Control Development Board User’s Guide” (DS70098)
- “dsPICDEM™ MC1L 3-Phase Low-Voltage Power Module User’s Guide” (DS70097)
- “PICDEM™ MC LV Development Board User’s Guide” (DS51554)

For information on the PID algorithm, see:

- “16-Bit Language Tools Libraries” (DS51456)

TROUBLESHOOTING

Problem: The motor ramps up and then, at mid to high-speed, it just quits.

Possible Solution: Try (a) increasing the blanking count; (b) if you have a lower speed motor (maximum speed much lower than 90,000 electrical RPM) and you are adding a lot of phase advance, you may need to have the algorithm transition from the low-speed implementation to the high-speed implementation earlier. Adjust the crossover point lower (from 250 to 200, then to 150). Decreasing the phase advance slope may also help.

Problem: The back-EMF from my motor does not look trapezoidal at all on the oscilloscope.

Possible Solution: This algorithm requires a trapezoidal shape to the motor's back-EMF signal. *AN1078, "Sensorless Field Oriented Control of PMSM Motors"* explains how to run a sinusoidal motor in Sensorless mode.

Problem: When running the code, an oscillator trap occurs.

Possible Solution: After programming the device with the debugger, make sure you reset the MPLAB ICD 2.

Problem: The MPLAB ICD 2 will not debug/program the device (when using the PICDEM MC LV board).

Possible Solution: Make sure S4 is in the PRGM position when programming the device. Switch S4 to the DEBUG position once the device is programmed and before debugging.

Problem: How do I know how fast the motor is going.

Possible Solution: Select "Watch" from the View menu in MPLAB ICD 2. This brings up the Watch window. Enter the variable "Speed" in the Watch window. Run the motor using the algorithm. When the motor is at the desired speed, press the "motor off" button. Halt the ICD 2. The speed of the motor at the time the "motor off" button was pressed will be displayed (in RPM) in the Watch window.

Problem: The motor you are using does not need to run faster than 40,000 electrical RPM and phase advance is not required.

Possible Solution: Set the crossover frequency to the maximum allowable value – 900 electrical revolutions per second. This will insure the motor only runs in Low-Speed mode. High-Speed mode is not necessary for motors that run slower than 40,000 electrical RPM. The number, 40,000 electrical RPM, was arrived at by adding up the processing and filter phase delays. The maximum amount of delay that can occur in Low-Speed mode is 30 electrical degrees. When the delays are added up (assuming no phase advance) and set equal to 30 electrical degrees, the fastest electrical speed is calculated to be around 48,000 RPM.

Problem: The motor stops running right after the start-up ramp. It seems like it should have enough speed to start.

Possible Solution: When first tuning the start-up ramp, make sure the potentiometer input and speed control loop are disabled. These can be disabled by the boolean controls on the Dynamic Data Control screen of the DMCI tool.

Problem: How do I know if the motor is running in High-Speed mode or not?

Possible Solution: Add the bit array, "ControlFlags", to the Watch window. The High-Speed mode bit is set when the motor is running in High-Speed mode. The bit is clear when the motor is in Low-Speed mode. You will have to halt the ICD 2 when the motor is running to check this bit.

Alternatively, check LED D5 if using the MC1 board – this LED is on when the motor is in High-Speed mode.

Problem: When I add phase advance, the motor does not cross over to High-Speed mode.

Possible Solution: Too much phase advance when the motor is running in Low-Speed mode will cause the motor to run improperly. Set the phase advance start frequency past the crossover frequency. The crossover frequency is in electrical revolutions per second. To convert to RPM, add 50 to the crossover frequency, multiply by 60 and divide by the number of pole pairs.

Problem: When using the MC1 board, I always get an oscillator trap when I first try to run the ICD 2 after I program the dsPIC30F6010A.

Possible Solution: After the dsPIC30F6010A is programmed, reset the ICD 2. Then, run the ICD 2 and halt the ICD 2. Next, click "run" on the DMCI tool. If another trap occurs, you will have to go through the sequence just described to clear the trap and load the parameters in the DMCI tool into the device.

Problem: I change the position of the sliders in the DMCI tool and it appears like the parameters are not updated in the dsPIC DSC.

Possible Solution: First, the DMCI tool is not a real-time tool (at the time of this writing). In order for changes you make to parameters to be loaded into the dsPIC DSC, you will have to halt the ICD 2, make the desired change to a parameter and restart the ICD 2.

Secondly, make sure the code is not stuck in a trap. To check to see whether the device is in a trap, halt the ICD 2 and look at the `trap.c` file. If the PC counter arrow (green arrow) is located in that file, then a trap has occurred. Clear the trap by resetting the ICD 2 and checking for other Fault sources.

Problem: How do I know how fast my motor is running?

Possible Solution: Add the “Speed” variable to the Watch window. When the motor is at speed, press the tactile switch that stops the motor. Halt the ICD 2. The Watch window will display the speed of the motor at the time you stopped the motor.

Problem: I am measuring the speed of the motor using a handheld tachometer. The value in “Speed” does not match my measured speed.

Possible Solution: Assuming the handheld tachometer is correct, make sure that the processor is running at the instruction cycle rate specified by `FCY` in `general.h`. In other words, if the processor is running at 30 MIPS and `FCY` is specified as 20000000, then the calculated motor speed will not be accurate.

Problem: I get an oscillator trap.

Possible Solution: After programming the device, reset the device before running the debugger.

Problem: Why do I have to run, and then halt the debugger once, before any adjustments to the motor tuning parameters in the DMCI tool will take effect?

Possible Solution: The motor start-up parameters are initialized in the source code. When you first run the debugger, the start-up parameters are initialized to the values specified in the source code (see `TuningInterface.c`). You must run the code beyond the initialization point for these parameters and then any adjustments you make to these parameters, using the DMCI tool, will take effect. The motor tuning parameters are initialized in the source code so that when you are finished tuning the motor, you can “hard code” the tuned parameters into the code and program the device.

AN1083

NOTES:

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.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELOQ, KEELOQ logo, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, PowerSmart, rPIC, and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AmpLab, FilterLab, Linear Active Thermistor, Migratable Memory, MXDEV, MXLAB, PS logo, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rLAB, rfPICDEM, Select Mode, Smart Serial, SmartTel, Total Endurance, UNI/O, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2007, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

**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 Mountain View, California. 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

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 - Fuzhou
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

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

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 - Shunde
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

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

China - Xian
Tel: 86-29-8833-7250
Fax: 86-29-8833-7256

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

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 - Gumi
Tel: 82-54-473-4301
Fax: 82-54-473-4302

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

Malaysia - Penang
Tel: 60-4-646-8870
Fax: 60-4-646-5086

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: 886-3-572-9526
Fax: 886-3-572-6459

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