INTRODUCTION

There is a lot of interest in using Brushless DC (BLDC) motors. Among the many advantages to a BLDC motor over a brushed DC motor are the following:

• The absence of the mechanical commutator allows higher speeds
• Brush performance limits the transient response in the DC motor
• With the DC motor, add the voltage drop in the brushes among motor losses
• Brush restrictions on reactance voltage of the armature constrains the length of core, reducing the speed response and increasing the inertia for a specific torque
• The source of heating in the BLDC motor is in the stator, while in the DC motor it is in the rotor, therefore it is easier to dissipate heat in the BLDC
• Reduced audible and electromagnetic noise

There are many different types of brushless motors, and the differences are:

• The number of phases in the stator
• The number of poles in the rotor
• The position of the rotor and stator relative to each other (rotor spinning inside the stator vs. rotor spinning outside the stator)

This application note discusses the three-phase motors. Two-phase motors are discussed in AN1178, “Intelligent Fan Control” (DS01178) while one-phase motors are a degenerated form of two-phase motors.

BACKGROUND

For a full description of three-phase brushless motors, read the application note “Brushless DC Motor Control Made Easy” (DS00857). AN857 is an excellent description of brushless motors and how to drive them with sensor feedback for commutation. With more advanced comparator modes and some new software techniques, this application note demonstrates an improved sensorless commutation strategy that has a much higher performance.
FIGURE 1: MOTOR POWER DRIVER
In this sample schematic, there are three P-Channel MOSFETs controlling the current flow from +Vcc into each phase. There are also three N-Channel MOSFETs controlling the current flow from each phase into ground. Between the N-Channel MOSFETs and ground there is a small resistor (R7) that allows the current through the motor to be sensed as a small voltage proportional to the current. Three BJT transistors are used to drive the P channel MOSFETs. The N channel MOSFETs are driven from the PIC® MCU I/O pins. For small MOSFETs and/or bipolar transistor output stages, MOSFET drivers are not required.

**Back EMF Sensing**

In order to learn the current position of the rotor, it is critical that some form of rotor position sensing is included. In a sensored design, the rotor position sensing is provided by a series of Hall effect sensors that react to the permanent magnetics in the rotor. For sensorless designs, the rotor position is set by knowing when a magnetic pole crosses the non-driven phase. During each commutation cycle, one phase is left undriven so it can sense the passing of a magnet on the rotor. The following circuit is self-biased and uses one comparator to perform the back EMF position sensing.

**FIGURE 2: BACK EMF SYSTEM**

Notice that the back EMF system consists of four elements with three of them repeating. The purpose of these elements is to detect the zero-crossing event even when the VDD voltages are changing. There are two easy ways to detect the middle of a sine wave. The first method is to make an inverted copy and compare them. The point where the two waves cross is the midpoint. The second method is to make a reduced amplitude copy and compare them. Again, the point where the two waves cross is the midpoint. The simplest method is the second, because it only requires a single comparator and a few resistors. Because this motor is a three-phase system, there are six zero-crossings per electrical rotation, the rising edge crossings and three falling edge crossings. When the commutation takes place, one of the three phase inputs is selected by writing to the CMxCON0 SFR in the microcontroller. To save cost, there is not a hardware filter on the comparator input, therefore, a noisy motor can cause false zero crossings. The solution is a software-based majority detector. To simplify this majority detector, the polarity bit in the CMxCON0 register is toggled with each commutation. Toggling the comparator output polarity with each commutation event makes all zero crossings look like a falling edge on the comparator output.

**Current Monitoring**

Current monitoring is a great feature for any motor control, but can be especially beneficial for BLDC motors. The benefits of current monitoring are:
- High current (no zero-crossings indicate a stuck rotor)
- Overcurrent limiting
- Torque control

Adding current monitoring is a simple task of inserting a small sense resistor in the ground return path of the half-bridge switching elements. An op amp may be necessary if the sense resistor is very small.

The simplest possible overcurrent monitor is to simply reset the microcontroller and restart commutation. This method is shown in Figure 1. The current sense resistor is used to drive the base of Q7. This transistor will cause a Reset of the microcontroller, if external MCLR is enabled. If external MCLR is not enabled, then the software can be extended to poll this input and take corrective action if an overcurrent condition is detected.

**SOFTWARE**

The software accomplishes the following tasks:
- Start the motor
- Detect zero crossing
- Commutate the stator
- Adjust commutation rate to match motor speed

**Starting the motor**

Starting the motor is the trickiest part of sensorless drives. The simplest method to start the motor is to simply start commutating at a slow rate and low duty cycle. The commutating should “catch” the rotor and, at some point, the zero-crossing detector will begin to see crossings. Once zero crossings can be measured, the rotor has begun rotating in synced with the commutation, and normal operation can begin. This method is very simple, but there are a few problems:
- The motor can spin erratically until sync is achieved
- The motor can sync at a harmonic of the actual speed
- It can take a long time for the motor to start up
To resolve these drawbacks, there are other methods that can be used to map the stalled position of the rotor and immediately start commutating from that point.

For many motors, the simple method of a time out on the zero crossing forcing a commutation will result in satisfactory performance; therefore, this is the method for this application note.

**Zero-Crossing Detector**

The zero-crossing system consists of switching the inputs to a comparator synchronously with the commutation and monitoring the output of the comparator. The comparator output is filtered with a majority detector. This filter is table-driven and looks for a transition from mostly 1’s to mostly 0’s. Once the transition is detected, the commutation can take place.

**Zero-Crossing Majority Detector**

In a noiseless system, zero-crossing events can be determined by observing when the output of a comparator sensing the back EMF voltage transitions from one to zero. Switching high currents at high voltages introduces a tremendous amount of noise into the system (see Figure 3). Determining when a zero-crossing event occurs in such an environment requires some sort of filtering to mitigate the noise. Filtering with discrete components adds too much delay to be useful, especially at high motor speeds. Discrete filters also vary with temperature, which adds to the complexity of delay management. A better filter is the one that has a predictable delay that does not vary with the environment.

A majority filter is one that can be implemented in software. Software filters have a predictable and fixed delay that is not affected by the environment. The filter uses a series of comparator output samples to detect a zero-crossing event. Zero crossing is said to have occurred when most of the first half of the samples are ones and most of the last half of the samples are zeros. For a six-sample window, a zero-crossing event is detected when two or three of the first three samples are ones and two or three of the last three samples are zeros. Table 1 illustrates all the possible combinations that satisfy these criteria.
TABLE 1: ZERO-CROSSING OCCURRENCES

<table>
<thead>
<tr>
<th>Bit Pattern</th>
<th>Numerical Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>011000</td>
<td>24</td>
</tr>
<tr>
<td>011001</td>
<td>25</td>
</tr>
<tr>
<td>011010</td>
<td>26</td>
</tr>
<tr>
<td>011100</td>
<td>28</td>
</tr>
<tr>
<td>101000</td>
<td>40</td>
</tr>
<tr>
<td>101001</td>
<td>41</td>
</tr>
<tr>
<td>101010</td>
<td>42</td>
</tr>
<tr>
<td>101100</td>
<td>44</td>
</tr>
<tr>
<td>110000</td>
<td>48</td>
</tr>
<tr>
<td>110001</td>
<td>49</td>
</tr>
<tr>
<td>110010</td>
<td>50</td>
</tr>
<tr>
<td>110100</td>
<td>52</td>
</tr>
<tr>
<td>111000</td>
<td>56</td>
</tr>
<tr>
<td>111001</td>
<td>57</td>
</tr>
<tr>
<td>111010</td>
<td>58</td>
</tr>
<tr>
<td>111100</td>
<td>60</td>
</tr>
</tbody>
</table>

The Most Significant bit of each bit pattern is the first sample of the series. As each new sample is taken, it occupies the Least Significant bit after all other bits are shifted left to make room. The Most Significant bit is dropped as a result of the shift. In effect, the bit pattern moves left through the six-sample window.

The majority filter is implemented in software by the following bits as they move through the window. Consider a sample window that starts with all zeros. When a logic high sample is taken, it is shifted left into the filter sample window. The resulting total value in the window becomes 1. As new samples are taken, they are shifted into the window, moving the existing samples left. If the first sample is one and all subsequent samples are zeros, the value in the window starts out as 1, then progresses to 2, 4, 8, 16, and finally 32, before it is shifted out and the window value returns to zero. The window value remains at zero until another logic high sample is taken. For each sample taken, the window value is first doubled and the logic level of the new sample is then added. For example, a window value that is 4 when a logic high sample is taken, becomes 8 plus 1 or 9. On the next sample, the 9 is then doubled by a left shift and the new sample is added, so that the result is either 18 or 19, depending on whether or not the new sample is a logic high.

At a first glance, one may think a majority filter can be constructed by using the sample window to address a look-up table. Addresses that match the majority criteria would return a zero-crossing indication flag from the table. This could work, except that some bit patterns will return multiple zero-crossing events as the pattern moves through the window. This could be solved by clearing the sample window after detecting an event. This has two problems: first, some patterns could never be reached and second, it takes time to clear the sample window.

For the first case, consider that pattern 60 will become either a pattern 56 or pattern 57 on the next sample, all of which will return the event flag. This suggests that there is a problem with the majority criteria table. Pattern 56 is actually a noiseless zero-crossing event and pattern 57 is a close second. With pattern 60 in the table, the real event pattern 56 cannot be reached. The simple solution is to remove pattern 60 from the table. This is not the only pattern with a problem. Pattern 28 will also become either pattern 56 or pattern 57 on the next sample. Pattern 28 also prevents pattern 56 from being reached. In fact, there are many other similar cases.

<table>
<thead>
<tr>
<th>Bit Pattern</th>
<th>Numerical Equivalent</th>
<th>Following Values</th>
<th>Preceding Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>011000</td>
<td>(24)</td>
<td>49*, 48*</td>
<td>44*, 12</td>
</tr>
<tr>
<td>011001</td>
<td>(25)</td>
<td>51, 50*</td>
<td>44*, 12</td>
</tr>
<tr>
<td>011010</td>
<td>(26)</td>
<td>53, 52*</td>
<td>45, 13</td>
</tr>
<tr>
<td>011100</td>
<td>(28)</td>
<td>57*, 56*</td>
<td>46, 14</td>
</tr>
<tr>
<td>101000</td>
<td>(40)</td>
<td>17, 16</td>
<td>52*, 20</td>
</tr>
<tr>
<td>101001</td>
<td>(41)</td>
<td>19, 18</td>
<td>52*, 20</td>
</tr>
<tr>
<td>101010</td>
<td>42</td>
<td>21, 20</td>
<td>53, 21</td>
</tr>
<tr>
<td>101100</td>
<td>44</td>
<td>25*, 24*</td>
<td>54, 22</td>
</tr>
<tr>
<td>110000</td>
<td>(48)</td>
<td>33, 32</td>
<td>56*, 24*</td>
</tr>
<tr>
<td>110001</td>
<td>(49)</td>
<td>35, 34</td>
<td>56*, 24*</td>
</tr>
<tr>
<td>110010</td>
<td>(50)</td>
<td>37, 36</td>
<td>57*, 25*</td>
</tr>
<tr>
<td>110100</td>
<td>52</td>
<td>41*, 40*</td>
<td>58*, 26*</td>
</tr>
<tr>
<td>111000</td>
<td>56</td>
<td>49, 48</td>
<td>60*, 28*</td>
</tr>
<tr>
<td>111001</td>
<td>57</td>
<td>51, 50*</td>
<td>60*, 28*</td>
</tr>
<tr>
<td>111010</td>
<td>58</td>
<td>53, 52*</td>
<td>61, 29</td>
</tr>
<tr>
<td>111100</td>
<td>(60)</td>
<td>57*, 56*</td>
<td>62, 30</td>
</tr>
</tbody>
</table>

Table 2 illustrates all the event values with values that precede and follow the event. Event values that are either preceded or followed by another event value should be considered for removal. The removal decision is based on which value best represents the actual zero-crossing event. Removing redundant values from the table also prevents skewing the zero crossing by inadvertent early detection of events. Events denoted by parentheses are covered by the preceding or following values denoted by an asterisk and, therefore, should be removed from the event table.
It may not be apparent why some event patterns are removed when one of the preceding values to that event is also removed. For example, event 50 has been removed because it is covered by the previous value 57. However, event 50 is not covered by the previous value 25, because that, too, has been removed. Event 25 was removed because it was covered by the previous event value 44 and non-event value 12. If event 25 remains in the table, it will trigger a false event after the previous value 12, therefore, it must go. Consequently, non-event 12 will propagate through value 25 and trigger event 50, if value 50 remains in the table. For that reason, event 50 must go. Similar arguments apply for the removal of values 49, 48, 41, and 40.

The look-up table is constructed by placing an event flag indicator at each address corresponding to a zero-crossing event. The flag is a special table value which will be discussed later. By filling all other locations of the table with double the relative address of the location truncated to six bits, a simple algorithm can be generated to work through the table as each bit is sampled. The algorithm adds the sample bit to the contents, at the previous table address, to create the new table address. If that new location contains the special flag, then the zero crossing has been detected and commutation action is taken.

The table contains 64 entries (addresses 0 through 63), since only six bits are used. The zero-crossing event flag is a value of one. Table entries with the value one then signal a zero-crossing event and temporarily set the next look-up address to one. This temporary address is cleared by the commutation routine so the sample window can start fresh looking for the next zero-crossing event. Table 3 illustrates the final majority filter table.

<table>
<thead>
<tr>
<th>Table Address</th>
<th>Table Contents</th>
<th>Table Address</th>
<th>Table Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>33</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>34</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>35</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>36</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>37</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>38</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>39</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>40</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>18</td>
<td>41</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>42</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>22</td>
<td>43</td>
<td>22</td>
</tr>
<tr>
<td>12</td>
<td>24</td>
<td>44</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>26</td>
<td>45</td>
<td>26</td>
</tr>
<tr>
<td>14</td>
<td>28</td>
<td>46</td>
<td>28</td>
</tr>
<tr>
<td>15</td>
<td>30</td>
<td>47</td>
<td>30</td>
</tr>
<tr>
<td>16</td>
<td>32</td>
<td>48</td>
<td>32</td>
</tr>
<tr>
<td>17</td>
<td>34</td>
<td>49</td>
<td>34</td>
</tr>
<tr>
<td>18</td>
<td>36</td>
<td>50</td>
<td>36</td>
</tr>
<tr>
<td>19</td>
<td>38</td>
<td>51</td>
<td>38</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>52</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>42</td>
<td>53</td>
<td>42</td>
</tr>
<tr>
<td>22</td>
<td>44</td>
<td>54</td>
<td>44</td>
</tr>
<tr>
<td>23</td>
<td>46</td>
<td>55</td>
<td>46</td>
</tr>
<tr>
<td>24</td>
<td>48</td>
<td>56</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>50</td>
<td>57</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>52</td>
<td>58</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>54</td>
<td>59</td>
<td>54</td>
</tr>
<tr>
<td>28</td>
<td>56</td>
<td>60</td>
<td>56</td>
</tr>
<tr>
<td>29</td>
<td>58</td>
<td>61</td>
<td>58</td>
</tr>
<tr>
<td>30</td>
<td>60</td>
<td>62</td>
<td>60</td>
</tr>
<tr>
<td>31</td>
<td>62</td>
<td>63</td>
<td>62</td>
</tr>
</tbody>
</table>

Table 3: Final Majority Filter Table
Commutation Phase Angle

The ideal commutation time is when the rotor magnets are 30 degrees away from the last zero-crossing point (see Figure 4). Since it takes a bit of time to energize the coils, a better commutation angle is often slightly early. To keep the system very simple, this application note uses 50% of the time between zero-crossings as the commutation point. This time corresponds to 30 degrees. It works well with many small motors.

The phase angle is computed as follows:

- Compute the 16 element rolling average of the commutation time.
- Divide the rolling average by 2.

The average acts as a low-pass filter and reduces jitter in the commutation timing. Excess jitter will increase current consumption and reduce the maximum speed.

Commutating

Commutating the motor is the simple task of controlling the comparator, PORT, and Core Independent Peripherals (CIPs) such as Capture/Compare/PWM module (CCP), Complementary Waveform Generator (CWG), and Complementary Output Generator (COG).

For the commutation sequence, please read the application note “Motor Control Made Easy” (DS00857). It contains several look-up tables that show which feedback to be read and which two of the outputs should be conducted in each commutation stage.

FIGURE 4: BLDC MOTOR WAVEFORM

The configurable CIPs and comparator are key elements to successful BLDC control with low-cost microcontrollers.

CORE INDEPENDENT PERIPHERALS (CIPS) USAGE

The PIC16 family offers powerful Core Independent Peripherals (CIPs) that can be very helpful to motor control applications. By using the CIPs, a lot of software overhead can be reduced, and complex but flexible signals can be generated. Therefore, the core can focus on the high-level control algorithm (in this case, the Majority Detector) of the motor, instead of generating the control signal. This section introduces some useful CIPs that are used in this application note, or can be used in general motor control applications.

Complimentary Waveform Generator (CWG)/Complimentary Output Generator (COG)

The CWG provides a complementary waveform with rising and falling edge dead-band control, enabling high efficiency synchronous switching, with no processor overhead. The CWG also incorporates auto-shutdown, auto-restart, and can interface directly with other peripherals/external inputs. The COG improves upon the functionality of the CWG with the addition of blanking and phase control.

In this application, the CWG/COG is used for modulating the 6-step control signal. The CWG/COG outputs are assigned to the three pins that control either the lower or the higher side of the half bridges. In the firmware, the output PWM can be easily switched to any of the three pins just by setting the output control register of the CWG/COG. Since the time it takes for switching are only caused by setting a register and the propagation delay in hardware, a faster 6-step switching can be achieved. The PIC16F153XX, PIC16F161X, and PIC16F188XX families all have at least one CWG/COG and are recommended for 3-phase BLDC trapezoidal control. For more available devices and information go to: http://www.microchip.com/design-centers/8-bit/peripherals/core-independent/complementary-waveform-generator-cwg.
24-bit Signal Measurement Timer (SMT)

The Signal Measurement Timer (SMT) is a 24-bit counter with advanced clocking and gating logic, which can be configured for measuring a variety of digital signal parameters such as pulse width, frequency and duty cycle, and the time difference between edges on two signals.

In many motor control applications, the SMT can be used for reading the speed reference signal, or processing the feedback signal in closed-loop controls. The SMT is available on PIC16F161X and PIC16F184XX families, which are recommended for motor control applications. For more available devices and information go to: http://www.microchip.com/design-centers/8-bit/peripherals/core-independent/24-bit-signal-measurement-timer-smt.

Stall Detection with Configurable Logic Cell (CLC) and Hardware Limit Timer (HLT)

The Configurable Logic Cell (CLC) generates customized combinational and sequential logic. The Hardware Limit Timer (HLT) is a timer mode in Timer 2/4/6 modules. It provides a timed hardware limit to be used in conjunction with asynchronous analog feedback applications, or to detect missed periodic events.

In almost all motor control applications, the CLC can be used to apply such logic that determines a stall condition, and then the HLT can be used for a fixed amount of trigger time. Therefore, a combination of CLC and HLT can generate a stall detection signal that indicates the motor has been stalled for a certain amount of time. The whole process is done without any core interference, so that there is no software overhead and there are no delays caused by interrupts or instructions in the software. The CLC and HLT are available in all the motor control recommended devices above. For more available devices and design tricks go to: http://www.microchip.com/design-centers/8-bit/peripherals/core-independent/configurable-logic-cell-clc, or HLT webpage: http://www.microchip.com/design-centers/8-bit/peripherals/core-independent/hardware-limit-timer-hlt.

CONCLUSION

The combination of flexible microcontroller features and majority filtering in software enables a sensorless 3-phase BLDC control system to be realized on a low-cost microcontroller. This implementation is ideal for cost-sensitive applications.

For reference designs and firmware solutions based on this topology, check the application note website http://www.microchip.com/wwwAppNotes/AppNotes.aspx?appnote=en534512. The firmware solution for each available device can also be found on the device’s front page.
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 unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KEELOG, KEELOG logo, Kleer, LANcheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntellIMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.


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

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

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

© 2008-2018, Microchip Technology Incorporated, All Rights Reserved.

ISBN: 978-1-5224-2626-4

Microchip received ISO/TS-16949:2009 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.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949

© 2008-2018 Microchip Technology Inc.
## 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://www.microchip.com/support](http://www.microchip.com/support)

**Web Address:**  
[www.microchip.com](http://www.microchip.com)

- **Atlanta**  
  Duluth, GA  
  Tel: 770-957-9614  
  Fax: 770-957-1455

- **Austin, TX**  
  Tel: 512-257-3370

- **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**  
  Novi, MI  
  Tel: 248-848-4000

- **Houston, TX**  
  Tel: 281-894-5983

- **Indianapolis**  
  Noblesville, IN  
  Tel: 317-773-8323  
  Fax: 317-773-5453  
  Tel: 317-536-2380

- **Los Angeles**  
  Mission Viejo, CA  
  Tel: 949-462-9523  
  Fax: 949-462-9608  
  Tel: 951-273-7800

- **Raleigh, NC**  
  Tel: 919-844-7510

- **New York, NY**  
  Tel: 631-435-6000

- **San Jose, CA**  
  Tel: 408-735-9110  
  Tel: 408-436-4270

- **Canada - Toronto**  
  Tel: 905-695-1980  
  Fax: 905-695-2078

### ASIA/PACIFIC

- **Australia - Sydney**  
  Tel: 61-2-9888-6733

- **China - Beijing**  
  Tel: 86-10-8569-7000

- **China - Chengdu**  
  Tel: 86-28-8665-5511

- **China - Chongqing**  
  Tel: 86-23-8980-9588

- **China - Dongguan**  
  Tel: 86-769-8702-9880

- **China - Guangzhou**  
  Tel: 86-20-8755-8029

- **China - Hangzhou**  
  Tel: 86-571-8792-8115

- **China - Hong Kong SAR**  
  Tel: 852-2943-5100

- **China - Nanjing**  
  Tel: 86-25-8473-2460

- **China - Qingdao**  
  Tel: 86-532-8502-7355

- **China - Shanghai**  
  Tel: 86-21-3326-8000

- **China - Shenyang**  
  Tel: 86-24-2334-2829

- **China - Shenzhen**  
  Tel: 86-755-8864-2200

- **China - Suzhou**  
  Tel: 86-186-6233-1526

- **China - Wuhan**  
  Tel: 86-27-5980-5300

- **China - Xiamen**  
  Tel: 86-3-8833-7252

- **China - Zhuhai**  
  Tel: 86-755-8864-2200

- **India - Bangalore**  
  Tel: 91-80-3090-4444

- **India - New Delhi**  
  Tel: 91-11-4160-8631

- **India - Pune**  
  Tel: 91-20-4121-0141

- **Japan - Osaka**  
  Tel: 81-6-6152-7160

- **Japan - Tokyo**  
  Tel: 81-3-6880-3770

- **Korea - Daegu**  
  Tel: 82-53-744-4301

- **Korea - Seoul**  
  Tel: 82-2-554-7200

- **Malaysia - Kuala Lumpur**  
  Tel: 60-3-7651-7906

- **Malaysia - Penang**  
  Tel: 60-4-227-8870

- **Philippines - Manila**  
  Tel: 63-2-634-9065

- **Singapore**  
  Tel: 65-6334-8870

- **Taiwan - Hsin Chu**  
  Tel: 886-3-577-8366

- **Taiwan - Kaohsiung**  
  Tel: 886-7-213-7830

- **Taiwan - Taipei**  
  Tel: 886-2-2508-8600

- **Thailand - Bangkok**  
  Tel: 66-2-694-1351

- **Vietnam - Ho Chi Minh**  
  Tel: 84-8-5448-2100

### EUROPE

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

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

- **Finland - Espoo**  
  Tel: 358-9-4520-820

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

- **Germany - Garching**  
  Tel: 49-8931-9700

- **Germany - Haan**  
  Tel: 49-2129-3766400

- **Germany - Heilbronn**  
  Tel: 49-7131-67-3636

- **Germany - Karlsruhe**  
  Tel: 49-721-025370

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

- **Germany - Rosenheim**  
  Tel: 49-8031-354-560

- **Israel - Ra'anana**  
  Tel: 972-9-744-7705

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

- **Italy - Padova**  
  Tel: 39-049-7625286

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

- **Norway - Trondheim**  
  Tel: 47-7289-7561

- **Poland - Warsaw**  
  Tel: 48-22-3325737

- **Romania - Bucharest**  
  Tel: 40-21-407-87-50

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

- **Sweden - Gothenburg**  
  Tel: 46-31-704-60-40

- **Sweden - Stockholm**  
  Tel: 46-8-5090-4654

- **UK - Wokingham**  
  Tel: 44-118-921-5800  
  Fax: 44-118-921-5820

© 2008-2018 Microchip Technology Inc.