INTRODUCTION
This application note describes a drive solution for the Single-Phase Brushless DC (BLDC) motor. The feature-rich peripherals of Microchip’s PIC16F1618 enable the control of the H-bridge switches, set motor speed, predict rotor position, correct speed error, detect fault events, optimize commutation and ensure program memory data integrity.

The solution described in this application note has the following key features:
• Full-Bridge Motor Control
• External PWM Speed Control
• Software-Configurable, Hardware-Implemented PID
• Overcurrent Detection
• Overtemperature Detection
• Motor Stall Detection
• Supports IEC60730 Standard Class B Certification for Invariable Memory Testing

Figure 1 shows the block diagram of a Single-Phase BLDC motor driver based on the PIC16F1618 microcontroller. The motor driver utilizes the Core Independent Peripheral (CIP) in the microcontroller to perform motor control function with minimum intervention from its Central Processing Unit (CPU). These are the CIPs used in the design:
• Signal Measurement Timer (SMT)
• Complementary Waveform Generator (CWG)
• Cyclic Redundancy Checking (CRC/Scan)
• Hardware Limit Timer (HLT)
• Math Accelerator (MathACC)
• Peripheral Pin Select (PPS)
• Temperature Indicator (TempIndi)
• Comparator (CMP)

Combining the CIPs with other on-chip peripherals, such as I/O ports, Analog-to-Digital Converter (ADC), Fixed Voltage Reference (FVR), Digital-to-Analog Converter (DAC), Pulse-Width Modulation (PWM) and Timers, brings the intelligence of the whole system.

These peripherals are internally connected by firmware, significantly reducing the number of external pins required for the implementation. Refer to Appendix A: “Circuit Schematic” for the detailed schematic diagram.
CONTROLLING THE MOTOR

Figure 2 shows the control diagram of the motor driver. The driver utilizes two feedback loops: an inner loop which is responsible for the motor electronic commutation control, and an outer loop which is responsible for speed control and correction.

FIGURE 2: CONTROL DIAGRAM

SPEED REFERENCE

The speed reference block seen in Figure 2 sets the desired speed of the motor. The higher the speed reference value, the faster the motor will spin. Calculation of the speed reference depends on the following parameters:

- Rated/Nominal Motor Speed
- Signal Measurement Timerx Captured Period Register (SMTxCPR) Value
- Signal Measurement Timerx Captured Pulse-Width Register (SMTxCPW) Value

The rated motor voltage and speed can be found in the technical specifications of the motor. For this application note, the rated motor voltage is 9V and the rated motor speed is 3200 RPM.

External PWM Signal Speed Reference

The speed reference of the motor driver is based on an external PWM signal. As an input to the motor controller, the duty cycle of the external signal determines the speed of the motor. The higher the duty cycle, the faster the motor will spin.

Measuring the input signal's duty cycle and converting it to the equivalent speed reference in Revolutions per Minute (RPM) are made possible through the use of the Signal Measurement Timer (SMT) peripheral. When the SMT is configured for Period and Duty Cycle mode with repeated data acquisition, the peripheral divides its clock source (SMTxCLK) by the input signal and captures the pulse width and period through the SMTxCPW and SMTxCPR registers, respectively. Equation 1 represents the equations used by the SMT to determine the value of SMTxCPR and SMTxCPW. Once the values of SMTxCPR and SMTxCPW are determined, they will be used in calculating the PWM duty cycle, as shown in Equation 2. The result will be multiplied by the rated motor speed to get the speed reference (RPM), as indicated in Equation 3. Using a 500 Hz input signal with HFINTOSC (16 MHz) as the selected SMT clock source, Table 1 shows the summary of the speed reference in relation to the input signal's duty cycle.
EQUATION 1:  SMT PULSE-WIDTH AND PERIOD CALCULATION

\[ SMT \times CPW = \frac{SMT \times CLK}{Input \ Signal} \times Duty \ Cycle(\%) \]

\[ SMT \times CPR = \frac{SMT \times CLK}{Input \ Signal} \]

EQUATION 2:  INPUT SIGNAL DUTY CYCLE CALCULATION

\[ Duty \ Cycle = \frac{SMT \times CPW}{SMT \times CPR} \]

\[ Duty \ Cycle(\%) = \frac{SMT \times CPW}{SMT \times CPR} \times 100 \]

EQUATION 3:  DUTY CYCLE TO SPEED REFERENCE CALCULATION

\[ Speed \ Reference (RPM) = Duty \ Cycle \times Rated \ Motor \ Speed \]

TABLE 1:  RPM VALUE BASED ON INPUT SIGNAL DUTY CYCLE

<table>
<thead>
<tr>
<th>Duty Cycle</th>
<th>SMTxCPW</th>
<th>SMTxCPR</th>
<th>Speed Reference (RPM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>32000</td>
<td>32000</td>
<td>3200</td>
</tr>
<tr>
<td>90%</td>
<td>28800</td>
<td>32000</td>
<td>2880</td>
</tr>
<tr>
<td>80%</td>
<td>25600</td>
<td>32000</td>
<td>2560</td>
</tr>
<tr>
<td>70%</td>
<td>22400</td>
<td>32000</td>
<td>2240</td>
</tr>
<tr>
<td>60%</td>
<td>19200</td>
<td>32000</td>
<td>1920</td>
</tr>
<tr>
<td>50%</td>
<td>16000</td>
<td>32000</td>
<td>1600</td>
</tr>
<tr>
<td>40%</td>
<td>12800</td>
<td>32000</td>
<td>1280</td>
</tr>
<tr>
<td>30%</td>
<td>9600</td>
<td>32000</td>
<td>960</td>
</tr>
<tr>
<td>20%</td>
<td>6400</td>
<td>32000</td>
<td>640</td>
</tr>
<tr>
<td>10%</td>
<td>3200</td>
<td>32000</td>
<td>320</td>
</tr>
</tbody>
</table>
INNER LOOP

In a Single-Phase BLDC motor, the four-slot stator contains the windings and the rotor is a four-pole permanent magnet. Figure 3 shows the standard structure of a Single-Phase BLDC motor. The rotor is the one that produces the rotating motion of the motor. To make the rotor spin, there must be a rotating magnetic field produced by the stator. The single winding is electrically activated to create a rotating field. To provide continuous rotation and prevent the permanent magnet rotor from getting locked with the stator, the excitation on the stator winding must be sequenced in a specific manner while knowing the exact position of the rotor magnets. The rotor magnet position is determined by using a Hall effect sensor.

**FIGURE 3: SINGLE-PHASE BLDC MOTOR STRUCTURE**

Figure 4 shows the timing control based on the Hall sensor. The Complementary Waveform Generator (CWG) output, which controls the excitation of the stator winding, is dependent on the state of the Hall sensor output. In order to control the CWG output, the Hall sensor output is compared to a Fixed Voltage Reference (FVR) by the comparator. The comparator hysteresis is enabled to disregard the noise that might add to the Hall sensor output. The output of the comparator toggles the CWG1MODE0 bit of CWG’s CWG1CON0 register. When the CWG1MODE0 bit is active, the CWG output is in Forward Full-Bridge mode and when inactive, the CWG output is in Reverse Full-Bridge mode. The toggling from Forward-to-Reverse mode produces a clockwise rotation, while toggling from Reverse-to-Forward mode produces a counter-clockwise rotation.

The CWG output is fed to the switches’ input of the full(H)-bridge circuit. Figure 5 shows the forward and reverse full-bridge operation and the corresponding magnetic polarity that the four-slot stator should produce based on the CWG output. In Forward mode, Q1 is on, Q2 and Q3 are off and Q4 is modulated. While in Reverse mode, Q1 and Q4 are off, Q2 is modulated and Q3 is on. In order to produce one electrical cycle, a forward-reverse combination must be executed. One mechanical revolution of the motor requires two electrical cycles, therefore, two forward-reverse combinations must be executed to complete a single clockwise rotation of the motor.
FIGURE 4: SENSOR AND DRIVE TIMING DIAGRAM

1 Mechanical Cycle (1 Rotation)
1 Electrical Cycle
1 Electrical Cycle

Comp/ Hall Sensor

CWG Input Source (PWM3) +V
GND

CWG1A +V
GND

CWG1B +V
GND

CWG1C +V
GND

CWG1D +V
GND

Motor Pin A -V
+V

Motor Pin B -V

CWG1MODE Forward Mode Reverse Mode Forward Mode Reverse Mode

0 90 180 270 0 90 180 270 0

+V
GND
+V
GND
GND
GND
GND
GND
+V
GND
+V
GND
- V
+V
- V
FIGURE 5: FORWARD AND REVERSE FULL-BRIDGE OPERATION

<table>
<thead>
<tr>
<th></th>
<th>FORWARD MODE</th>
<th>REVERSE MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>Q2</td>
<td>OFF</td>
<td>MODULATED</td>
</tr>
<tr>
<td>Q3</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>Q4</td>
<td>MODULATED</td>
<td>OFF</td>
</tr>
</tbody>
</table>
Full(H)-Bridge Circuit

The schematic diagram of the full-bridge circuit, seen in Appendix A: “Circuit Schematic”, is primarily composed of two P-channel MOSFETs as high-side switches and two N-channel MOSFETs as low-side switches. The main advantage of using a P-channel MOSFET, compared to an N-channel MOSFET as a high-side switch, is that the gate driving technique for the high-side switch position is simpler, thus reducing the cost of the high-side gate driving circuit. Transistor Q5 and resistors R1, R3 and R5 are configured as an emitter-follower that acts as a level shifter circuit to drive P-channel MOSFET Q1. Similarly, on the other arm of the full-bridge circuit, transistor Q6 and resistors R2, R4 and R5 are configured to drive Q3 in this manner. When the CWG1A output is high, Q5 is conducting and the voltage across R1 relative to V_MOTOR provides negative voltage to charge the gate-to-source capacitance, and turns on Q1. Likewise, when CWG1C is high, Q6 and the voltage across R2 works in the same manner that turns on Q3. The voltage across R1 and R2 when turning on Q1 and Q3 should be greater than the Gate Threshold Voltage VGS(TH), and lower than the Maximum Gate-to-Source Voltage VGS(MAX) of Q1 and Q3. The voltage across the gate-to-source of Q1 and Q3 can be calculated using Equation 4 and Equation 5. When CWG1A is low, Q5 is off and the gate-to-source capacitance voltage of Q1 discharges through R1, and turns off Q1. Likewise, when the CWG1C is low, Q6 and the Q3 discharge on R2 works in the same manner that turns off Q3. Q1 and Q3 effectively invert the CWG1A and CWG1C PWM output signal.

Cross Conduction, the condition where the high and low-side switches are both switched on, should be avoided. Otherwise, it will create a current shoot-through that might damage the driver’s components. Using the CWG’s Counter registers, a dead-band delay can be imposed on the CWG outputs. This provides non-overlapping output signals that will not allow the high and low-side switches to conduct at the same time. The CWG contains two 6-bit dead-band delay counters, one for the rising edge of the input source and the other for the falling edge of the input source. This dead-band delay is timed by counting CWG clock periods from zero up to the specified value in the two CWG Counter registers (CWG1DBR and CWG1DBF). Figure 6 depicts the CWG outputs with dead-band delay.

**EQUATION 4:**  
**P-CHANNEL MOSFET (Q1) GATE-TO-SOURCE VOLTAGE (Vgs)**

\[ V_{GS(Q1)} = -\left(\frac{V_{DD} - V_{BE(Q1)}}{R_3} \cdot \frac{R_1}{\beta_{DC}} + \frac{R_5}{\beta_{DC}}\right) \times \left(1 - \frac{1}{\beta_{DC}}\right) \]

\[ \beta_{DC} = Q5 \text{ DC Current Gain} \]

**EQUATION 5:**  
**P-CHANNEL MOSFET (Q3) GATE-TO-SOURCE VOLTAGE (Vgs)**

\[ V_{GS(Q3)} = -\left(\frac{V_{DD} - V_{BE(Q3)}}{R_4} \cdot \frac{R_2}{\beta_{DC}} + \frac{R_5}{\beta_{DC}}\right) \times \left(1 - \frac{1}{\beta_{DC}}\right) \]

\[ \beta_{DC} = Q6 \text{ DC Current Gain} \]
Although dead bands are already provided, there’s still a possibility that Q1 and Q2 or Q3 and Q4 conduct at the same time. This is because of the self turn-on phenomenon of the MOSFET related to the gate-to-drain internal miller capacitance. When Q1 is off and Q2 is modulating, at the instant Q2 turns on, the drain voltage of Q1 and Q2 drops from V_MOTOR to 0V. The rapidly falling change of voltage (dv/dt) at the drain of Q1 produces current, via a parasitic Gate-to-Drain Miller Capacitor (C_GD) of Q1, to flow in R1 and the internal Gate-to-Source Capacitor (C_GS) of Q1. As a result, V_GS of Q1 can increase from 0 to a certain voltage level while it is in its OFF state. If V_GS reaches the threshold voltage, V_TH of Q1, Q1 will falsely turn on and cross conduction of Q1 and Q2 will happen. Similarly, when Q1 is off and Q2 is off, at the instant Q1 turns on, the drain voltage of Q1 and Q2 rises from 0 to V_MOTOR. The rapidly rising change of voltage (dv/dt) at the drain of Q2 produces current, via the C_GD of Q2, to flow through R6 and the C_GS of Q2. As a result, the V_GS of Q2 can increase from 0 to a certain voltage level while it is in its OFF state. If the V_GS reaches the V_TH of Q2, Q2 will falsely turn on and cross conduction of Q1 and Q2 will happen. The other arm of the full-bridge circuit is also susceptible to the occurrence of this phenomenon.

The coupling effect at the V_GS can be roughly calculated using Equation 6, where R is the total gate resistance in the circuit, C_RSS is equal to C_GD, C_ISS is equal to C_GS + C_GD, V_DS is the Drain-to-Source Voltage of the low-side switch and dv/dt is the drain-to-source voltage change of the low-side switch.

**EQUATION 6: V_GS COUPLING EFFECT CALCULATION**

\[
V_{GS(MILLER)} = \frac{dv}{dt} \times R \times C_{RSS} \left\{ \frac{-V_DS}{\frac{dv}{dt} \times R \times C_{ISS}} \right\} \left( 1 - e^{-\frac{V_DS}{\frac{dv}{dt} \times R \times C_{ISS}}} \right)
\]
In order to reduce the internal coupling effect, an external capacitor is connected across the gate-to-source terminal of the MOSFETs. In the schematic diagram, C1, C2, C3 and C4 are connected across the gate-to-source terminals of Q1, Q2, Q3 and Q4, respectively. These capacitors increase the effective Ciss for each respective MOSFET. For example, when C1 is connected across the gate-to-source of Q1, the effective value of the Ciss of Q1 is equal to Cgs + Cgd + C1. In reference to Equation 6, when Ciss increases, the Vgs of Q1, due to Miller coupling, decreases.

In this application, the key parameters in selecting MOSFETs are based on the device Rds(on) (On-Resistance) and Qg (Total Gate Charges). Ideally, the N-channel and P-channel MOSFETs in the full-bridge circuit should have the same Rds(on) and Qg in order to attain the optimal switching performance. Therefore, it is convenient to choose a complementary pair of a P-channel and an N-channel device in order to match these parameters. However, this is impossible due to the difference of construction of the two MOSFETs. The chip size of the P-channel must be two to three times that of the N-channel in order to match the N-channel Rds(on) performance, but the larger the chip size, the larger the effect on its dynamic performance, such as Qg. On the other side, when the P-channel has the same Qg as the N-channel, the P-channel and N-channel have the same chip size, but the P-channel has larger Rds(on) than the N-channel. Because of this trade-off, it will be necessary to choose which parameter between Rds(on) and Qg will mostly affect the switching performance of the MOSFET. The decision is determined based on the MOSFET’s operating switching frequency. At high-frequency operation (greater than 50 kHz), switching losses are dominant. When the Qg of P-channel is comparable with the N-channel MOSFET, it will significantly reduce the total MOSFET power losses. Otherwise, at low-frequency operation (less than 50 kHz), conduction losses are dominant. When Rds(on) of the P-channel is comparable with the N-channel MOSFET, it will significantly reduce the total MOSFET power losses. In this motor driver solution, the switching frequency is 15.625 kHz; therefore, the P-channel and N-channel MOSFETs were chosen through their comparable Rds(on), as shown in Table 2.

### Table 2: MOSFET Characteristics (Qg AND Rds(on))

<table>
<thead>
<tr>
<th>MOSFET Type</th>
<th>P/N</th>
<th>Qg</th>
<th>Rds(on)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-Channel</td>
<td>FDS6375</td>
<td>26 nC</td>
<td>0.024Ω at Vgs = -4.5V</td>
</tr>
<tr>
<td>N-Channel</td>
<td>NDS8425</td>
<td>11 nC</td>
<td>0.022Ω at Vgs = 4.5V</td>
</tr>
</tbody>
</table>

**Note:** An alternate Micrel MOSFET driver can also be used with the designed full-bridge circuit in this application note. Refer to [www.micrel.com](http://www.micrel.com) for the complete list of available full-bridge drivers.
OUTER FEEDBACK LOOP

The outer loop, shown in Figure 2, provides measurement and control of the motor's speed. The objective in controlling the motor's speed is to maintain the speed of the motor at the desired value under various conditions, such as change in load demand, disturbances and temperature drift. To implement an outer loop control, the motor's actual speed should be determined by applying necessary speed correction to maintain the desired value.

Actual Speed Measurement

Aside from calculating the desired speed reference, SMT is also used to measure the actual motor speed response. Measuring the motor's actual speed is done through the use of SMT Period and Duty Cycle Acquisition mode with repeated data acquisition. By enabling this mode, the peripheral will automatically measure the period of the input signal (SMTxIN) relative to the SMTx clock source (SMTxCLK) and the result will be stored in the SMTx Captured Period register (SMTxCPR), as shown in Equation 7. By combining Equation 7 and the general formula of the BLDC motor for calculating RPM, shown in Equation 8, the actual speed of the motor in RPM can be calculated as demonstrated in Equation 9.

EQUATION 7: SMT CAPTURED PERIOD REGISTER CALCULATION

\[ SMT \times CPR = \frac{SMT \times CLK}{Motor \ Frequency} \]

EQUATION 8: MOTOR RPM CALCULATION

\[ RPM = \frac{120 \times Motor \ Frequency}{Number \ of \ Poles} \]

EQUATION 9: COMBINED EQUATION TO CALCULATE ACTUAL SPEED

\[ Actual \ Speed \ (RPM) = \frac{120 \times SMT \times CLK}{Number \ of \ Poles \times SMT \times CPR} \]

Proportional-Integral-Differential Controller (PID)

The speed reference described in Equation 3 and actual speed described in Equation 9 are now compared to determine the error. This error can be positive or negative, which indicates that the actual speed is higher or lower than the speed reference. This error is fed to a PID controller to calculate a value that compensates the variation in speed.

The PID controller is an algorithm that provides stability in a system. It is usually implemented through software, which takes several instructions in performing a series of operation and iteration. In contrast with the usual implementation of the PID, the PID controller used in this design uses the on-chip MathACC with PID mode module of the PIC16F1618. This hardware-implemented PID performs the calculation very efficiently in the discrete time domain.

Figure 7 shows that the calculated speed reference is used as the PIDx Set Point (PIDxSET) while the actual speed of the motor is used as the PIDx Input (PIDxIN). The module will automatically calculate the compensating value and yield it as a PIDx Output (PIDxOUT). This value will be used to adjust the duty cycle of the CWG PWM output.
Example 1 shows the sample code for implementing the hardware PID calculation. When this routine is implemented in software it takes 100 instruction cycles or more, depending on the coefficient (Kp, Ki, Kd) used in tuning the PID. While using MathACC in PID mode, it takes only nine instruction cycles to perform the calculation. Refer to the technical brief TB3136, “PID Control on PIC16F161X by using a PID Peripheral” for more information on how to implement and use the PID Control mode features of the MathACC peripheral.

**EXAMPLE 1: CODE FOR IMPLEMENTING PID**

```c
void MATHACC_Initialize(void)
{
    PID1CONbits.EN = 1; // PID Module Enabled
    PID1CONbits.MODE = 0b101; // PID Mode Controller

    // Calculated K1, K2, K3 based on Kp, Ki, Kd and the sampling rate
    PID1K1 = 80;
    PID1K2 = -70;
    PID1K3 = 10;

    // Input Values Selection
    PID1SET = 0x00; // Desired Set Point
    PID1IN = 0x00; // Measured Process Variable

    while (PID1CONbits.PID1BUSY == 1)
    {
        result = PID1OUT; // Load the result
    }
}```
MOTOR DRIVER PROTECTION FEATURE

In order to avoid system failure, damage or motor driver performance degradation, appropriate early Fault detection strategies are implemented in this application.

Overcurrent Detection

Exceeding the motor’s maximum allowable torque loading can cause the motor winding to take the maximum allowed current, which may cause the motor to rise above its allowed operating temperature. To protect the motor from overheating, Fault detection for overcurrent must be implemented.

To implement overcurrent detection, an R SHUNT is added to the drive circuitry that gives a voltage corresponding to the current flowing in the motor winding. The voltage drop across this resistor varies linearly with respect to the motor current. The voltage is fed to the inverting input of the comparator and compared to a certain reference voltage. This reference voltage is based on the result between the RSHUNT resistance and the maximum allowable stall current of the motor. The reference voltage can be provided by the FVR, which can be narrowed down further by the DAC. In this manner, a very small reference voltage can be used, allowing the RSHUNT resistance to be kept low. Keeping the resistance low reduces the RSHUNT power dissipation. If the RSHUNT voltage exceeds the reference, the comparator output will trigger the auto-shutdown feature of the CWG.

Overtemperature Detection

Overtemperature can be detected using the device on-chip temperature indicator peripheral present within the PIC16F161X family. The indicator measures the device temperature, corresponding to the temperature in its environment with some delay.

The indicator is used to measure the device temperature between -40°C and +85°C. The internal circuit of the temperature indicator produces a variable voltage relative to temperature using an internal transistor junction threshold voltage. This voltage is converted to a digital form by the Analog-to-Digital Converter (ADC). The ADC result will be used to determine the actual temperature reading defined by Equation 10. For a more accurate temperature indicator reading, a single-point calibration is implemented. Refer to the application note AN1333, “Use and Calibration of the Internal Temperature Indicator” for more details regarding the calibration process.

EQUATION 10: TEMPERATURE READING CALCULATION

\[
\text{Temperature Reading} = 0.659 - \frac{V_{DD}}{2^n - 1} \left[ 1 - \frac{\text{ADC_RESULT}}{2^n} \right] - 40
\]

Note: High-Range mode = 4
     Low-Range mode = 2
     \( n \) = Number of bits of ADC Resolution
     \( \text{ADC_RESULT} \) = ADRES Register Value
The implementation of the overtemperature detection uses the ADC internal Channel Input Selection (CHS) bit. The Temperature Indicator (TempInd) module is used as the channel input for the ADC. For every timer interrupt, the completed ADC conversion result will be compared to the desired maximum temperature limit. When the ADC result exceeds the maximum temperature limit, the output of the CWG disables.

Table 3 summarizes the ADC result that the temperature indicator produces relative to the temperature.

**TABLE 3: TEMPERATURE INDICATOR ADC RESULT SUMMARY**

<table>
<thead>
<tr>
<th>Temperature (°C)</th>
<th>VTEMP mV</th>
<th>8-Bit ADC Result</th>
<th>10-Bit ADC Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>-40</td>
<td>2,364</td>
<td>121</td>
<td>484</td>
</tr>
<tr>
<td>-30</td>
<td>2,4168</td>
<td>123</td>
<td>494</td>
</tr>
<tr>
<td>-20</td>
<td>2,4696</td>
<td>126</td>
<td>505</td>
</tr>
<tr>
<td>-10</td>
<td>2,5224</td>
<td>129</td>
<td>516</td>
</tr>
<tr>
<td>0</td>
<td>2,5752</td>
<td>131</td>
<td>527</td>
</tr>
<tr>
<td>10</td>
<td>2,628</td>
<td>134</td>
<td>538</td>
</tr>
<tr>
<td>20</td>
<td>2,6808</td>
<td>137</td>
<td>548</td>
</tr>
<tr>
<td>30</td>
<td>2,7336</td>
<td>139</td>
<td>559</td>
</tr>
<tr>
<td>40</td>
<td>2,7864</td>
<td>142</td>
<td>570</td>
</tr>
<tr>
<td>50</td>
<td>2,8392</td>
<td>145</td>
<td>581</td>
</tr>
<tr>
<td>60</td>
<td>2,892</td>
<td>147</td>
<td>592</td>
</tr>
<tr>
<td>70</td>
<td>2,9448</td>
<td>150</td>
<td>603</td>
</tr>
<tr>
<td>80</td>
<td>2,9976</td>
<td>153</td>
<td>613</td>
</tr>
<tr>
<td>85</td>
<td>3,024</td>
<td>154</td>
<td>619</td>
</tr>
</tbody>
</table>

**Motor Stall Detection**

When the motor is spinning, the Hall sensor provides periodic pulses. If the motor stalls, the Hall sensor also stops from providing the pulses. The controller uses the Hall sensor response to detect the motor stalled condition. To implement the motor stall detection, the edge-triggered Hardware Limit Timer (HLT) mode in the Timer2/4/6 peripheral is used. Its main function is to monitor any changes in a periodic signal and is effectively set up as a retrigerable monostable. Refer to Figure 8 for HLT implementation on motor control design.
In this application, the Comparator 1 output, which is connected to the Hall sensor signal, is set as the external signal source for the HLT. The Hall sensor produces signals ranging from 32 Hz (31.25 mS) up to 110 Hz (9 mS) depending on the motor speed. The Period (PR2) register of the Timer2 is then set to a value that is sufficiently larger than the minimum input frequency (32 Hz). By doing this, motor stall condition can be detected at a much wider speed range. Refer to Equation 11 for the PR2 value calculation based on the Hall sensor’s signal.

**EQUATION 11: PR2 CALCULATION**

\[
PR2 > \frac{TMR2 \text{ Clock Source}}{Input \text{ Signal} \text{ (MIN) \times Prescaler \times Postscaler}}
\]

\[
PR2 > \frac{LFINTOSC}{35 \text{ Hz}} > \frac{31000}{32 \times 1 \times 8} > 121
\]

**Note:** PR2 register value should be greater than 121.

The reason for a larger PR2 value is for the Hall sensor pulse to occur first and reset the Timer2 count before the PR2 period match occurs. In case of a rotating motor, Hall sensor pulses are always present to continuously reset the Timer2 count, avoiding the period match to occur. Otherwise, in case of a stalled motor, Hall sensor pulses are not present to reset the Timer2 count, hence the timer continues to increment until the period match occurs. An interrupt event is triggered every time a period match occurs. This can be used to shut down the CWG output and indicate that the motor is not spinning or is in a stalled condition. Refer to Figure 9 for the implementation of HLT mode as stall detection.

**FIGURE 9: STALL DETECTION IMPLEMENTATION**

The diagram illustrates the difference between the motor normal running condition and the motor stall condition. In the normal running condition, the Hall sensor signal is present, and the Timer2 count resets at each rising edge, avoiding a period match. In the stall condition, the Hall sensor signal is not present, and the Timer2 count continues to increment until a period match occurs, triggering an interrupt event.

![Diagram showing Hall sensor, PR2 value, Timer clock, Timer count, and Timer interrupt for normal and stall conditions.](image-url)
Program Memory Data Monitoring and Checking

In a motor control application, data integrity is necessary to ensure the safe operation of the motor and all the Fault detection features are functioning reliably. A failure in the implementation of the motor function and all its safety features, due to corrupted program memory, may result in the damage of the drivers and of the control circuitry. To avoid this scenario, an error detection technique can be added in the control system to monitor the program memory data integrity. Using the Cyclic Redundancy Check (CRC) peripheral, along with the built-in Program Memory Scan module, a memory test procedure can be performed periodically during system run time without conflicting with the motor’s operation. This hardware CRC periodic memory test implementation is also helpful in complying with the IEC 60730 standard Class B certification in invariable memory (Flash/EEPROM) test. Refer to the application note AN1799, "Class B Safety Software Library for 8-bit PIC® MCUs", for more details regarding the program memory checking compliance procedure according to the IEC-60730 standard Class B certification.

The CRC peripheral provides a means for calculating a check value (checksum) of the program memory. By using the built-in Program Memory Scanner module, program memory can be converted into equivalent binary data and used as input data (CRCDAT) to the CRC. The CRC checksum (CRCACC) is generated by dividing the CRC input data (CRCDAT) by another binary number, called the CRC polynomial (CRCXOR). The CRC polynomial can be any polynomial up to 17 bits, or any commonly used standard polynomials, such as CRC-16-ANSI/0x8005, CRC-8/0xD5 and CRC-16-CCITT/0x1021. Figure 10 shows the simplified block diagram of CRC implementation in this application. Refer to the technical brief TB3128, “CRC and Memory Scan on 8-Bit Microcontrollers Technical Brief” for more details regarding the implementation of CRC and memory scan.

FIGURE 10: CRC AND MEMORY SCAN SIMPLIFIED BLOCK DIAGRAM
At the first system start-up, an initial memory scan is performed at the whole program memory address using the Burst mode operation. The Burst mode produces the highest scanner throughput at the expense of stalling other CPU functions. Therefore, it is usually used only during start-up. Scanner output will be used by the CRC to calculate the reference checksum value. Then, during the system run time, the scanner and the CRC calculation are periodically called to generate another checksum value of the same program memory address. The scanner scans in Peek mode to avoid the stalling of other CPU functions, but it produces the lowest scanner throughput. If the reference and the periodically generated checksum values match, an indicator can be set by the user to indicate that the program memory has passed the test and no errors were found. In case of a failed program memory testing, or in case errors were found, the CWG output shutdown will be triggered. Figure 11 shows the diagram of the CRC calculation routine used in this application.

FIGURE 11: CRC AND MEMORY SCAN IMPLEMENTATION ROUTINE
Figure 12 shows the motor driver firmware flowchart. During system start-up, the firmware initializes the peripherals and the connection amongst each other. After peripherals are initialized, the firmware enables the program memory scan and runs a CRC calculation to determine the initial checksum value of the program memory. This value is used as a reference and periodically compared with another checksum value, generated during system run time, to check and ensure the integrity of the program memory data.
After all the initialization, the firmware will enter a continuous loop, allowing the execution of the following repeated tasks:

1. Monitors the changes in the duty cycle of the input PWM signal and updates the speed reference setting. This task is performed by the SpeedReference_Calculation() routine.

2. Monitors the changes in the actual motor speed response and updates the speed reading. This task is performed by the ActualSpeed_Calculation() routine.

3. Monitors every Timer1 overflow interrupt and signals the firmware to start the PID calculation. Timer1 overflow corresponds to the desired PID controller sampling time and is equivalent to 10ms for this application.

**FIGURE 13: TIMER3, COMPARATOR 1 AND COMPARATOR 2 FLOWCHART**
Additionally, the firmware executes several Interrupt Service Routines (ISRs) indicating a certain motor condition. These interrupt handlers are automatically executed when certain peripheral criteria have been met. Refer to Figure 13 and Figure 14 for the peripheral ISR flowchart.

1. **Comparator 1 ISR** – Monitors the motor’s Hall sensor signal. This is executed every change of Hall sensor output state. When the Hall sensor signal is high, the CWG Forward mode is enabled, otherwise, the CWG Reverse mode is enabled.

2. **Comparator 2 ISR** – Monitors the system overcurrent. This is executed on every rising edge of the inverting input (C2IN1-) pin of the comparator connected to the sensing resistor (RSHUNT). When set, the CWG output shutdown is triggered.

3. **Timer3 ISR** – Monitors the system overtemperature. This is executed every Timer4 overflow or every 300s. It signals the ADC to calculate the equivalent binary data of the on-chip temperature indicator peripheral reading. Exceeding an 85°C reading will trigger a CWG output shutdown.

4. **Timer4 ISR** – Monitors the program memory data integrity. This is executed every Timer6 overflow or every 600s. It signals the Program Memory Scanner module and the Cyclic Redundancy Check (CRC) peripheral to calculate a CRC checksum value, and compare it to the previously generated checksum value during system start-up. A CWG output shutdown is triggered upon an unequal comparison.

5. **HLT Timer2 ISR** – Monitors the motor stall condition. This is executed on every PR2 match. When set, the CWG output shutdown is triggered.

Comparator 2, Timer3, Timer4 and the HLT ISR handle the system Fault condition. When any one of them is executed, the LED indicator turns on and the operation of the motor driver is terminated. The driver will restart its operation after resetting the MCU device and the Fault is not existing.

All peripherals used in the firmware are configured and initialized using the MPLAB® Code Configurator (MCC). Appendix B: “MPLAB® Code Configurator (MCC) Peripheral Initialization” provides the procedures on how the peripherals are initialized using MCC. For the complete source code, refer to Appendix C: “Source Code Listing”.

---

**FIGURE 14:  HLT AND TIMER4 ISR FLOWCHART**
MOTOR DRIVER PERFORMANCE

Figure 15 shows the motor speed step response.

FIGURE 15: PIC16F1618 MOTOR DRIVER SPEED STEP RESPONSE
CONCLUSION

In cost-sensitive motor control applications, an efficient and flexible microcontroller can have a significant impact. Device efficiency can be measured against the level of integrated peripheral optimized to control tasks along with the number of pins/memory and the size of the package. In addition, ease of use and less time to market are important, especially if multiple versions of the designs are required. This application note describes how a low-cost microcontroller meets these requirements. By using the PIC16F1618 microcontroller, the motor driver can control the driver switches, set motor speed, predict rotor position, correct speed error, detect Fault event, and ensure program memory data integrity.
APPENDIX A: CIRCUIT SCHEMATIC

FIGURE A-1: SENSORED SINGLE-PHASE BLDC MOTOR DRIVER USING PIC16F1618
SCHEMATIC DIAGRAM CIRCUIT
APPENDIX B: MPLAB® CODE CONFIGURATOR (MCC) PERIPHERAL INITIALIZATION

In this section, the MPLAB® Code Configurator (MCC) is utilized to easily configure the peripherals used in this motor control application. The MCC is a user-friendly plug-in tool for MPLAB® X IDE which generates drivers for controlling and driving peripherals of PIC® microcontrollers, based on the settings and selections made in its Graphical User Interface (GUI). Refer to the “MPLAB® Code Configurator User’s Guide” (DS40001725) for further information on how to install and set up the MCC in MPLAB X IDE.

The following figures will provide a guide on how to configure the PIC16F1618 peripherals in this application note using MCC.

FIGURE B-1: CONFIGURATION REGISTERS SETTINGS

![Figure B-1: Configuration Registers Settings](image1)

FIGURE B-2: SIGNAL MEASUREMENT TIMER1 (SMT1) CONFIGURATION FOR SPEED REFERENCE IMPLEMENTATION

![Figure B-2: Signal Measurement Timer1 Configuration](image2)
FIGURE B-3: PWM3 AND TIMER6 PERIPHERAL CONFIGURATION AS INPUT SOURCE FOR CWG

Please refer to the selected timer to adjust the PWM frequency

- Enable PWM
- PWM Polarity: ☐ inverted ☑ not inverted
- Select a timer: Timer6
- PWM Period: 256.000 µs
- Duty Cycle: ☐ 50 ☑ %
- CCFR Value: 511
- PWM Frequency: 3.9062 kHz
- PWM Resolution: 10 bits

FIGURE B-4: CWG CONFIGURATION FOR FULL-BRIDGE MOTOR DRIVER

- Enable CWG
- Input Source: PWM3_OUT
- Output Mode: Full bridge mode
- Clock: HFINTOSC (15 MHz)
- Dead-band Uncertainty:

- Rising Event:
  - Rising Counts: 80 to 15 counts
  - Minimum Dead-band: 62.5 ns
  - Maximum Dead-band: 512.5 ns

- Falling Event:
  - Falling Counts: 90 to 10 counts
  - Minimum Dead-band: 62.5 ns
  - Maximum Dead-band: 625.0 ns

- CWGA
  - Output Polarity: ☐ Inverted ☑ Non Inverted
- CWGB
  - Output Polarity: ☐ Inverted ☑ Non Inverted
- CWGC
  - Output Polarity: ☐ Inverted ☑ Non Inverted
- CWGD
  - Output Polarity: ☐ Inverted ☑ Non Inverted
FIGURE B-5: SIGNAL MEASUREMENT TIMER2 (SMT2) CONFIGURATION FOR ACTUAL SPEED MEASUREMENT IMPLEMENTATION

FIGURE B-6: MathACC AND TIMER1 CONFIGURATION FOR PID CONTROLLER
FIGURE B-7: DIGITAL-TO-ANALOG CONVERTER (DAC) CONFIGURATION

Enable DAC
Enable output on DAC1OUT1
Positive Reference: FVR

FIGURE B-8: FIXED VOLTAGE REFERENCE (FVR) AND DIGITAL-TO-ANALOG CONVERTER (DAC) CONFIGURATION FOR OVERCURRENT REFERENCE

Enable FVR
FVR amount sent to ADC: off
Enable Temperature Sensor
Voltage Range Selection: Lo_range
FVR amount sent to Comparators, DAC and CPS: 2x

FIGURE B-9: COMPARATOR 1 CONFIGURATION FOR HALL SENSOR REFERENCE

Enable Comparator
Enable Synchronous Mode
Enable Low Power
Enable Comparator Hysteresis
Enable Comparator Interrupt
Positive Input: FVR_pin
Negative Input: CIN1-
Output Polarity: Inverted, Non Inverted
Interrupt Flag Set On: rising edge, falling edge

FIGURE B-10: COMPARATOR 2 CONFIGURATION FOR OVERCURRENT DETECTION

Enable Comparator
Enable Synchronous Mode
Enable Low Power
Enable Comparator Hysteresis
Enable Comparator Interrupt
Positive Input: DAC_pin
Negative Input: CIN2-
Output Polarity: Inverted, Non Inverted
Interrupt Flag Set On: rising edge, falling edge
FIGURE B-11: CYCLIC REDUNDANCY CHECK (CRC) CONFIGURATION FOR PROGRAM MEMORY INTEGRITY CHECKING

FIGURE B-12: TIMER4 CONFIGURATION FOR PROGRAM MEMORY INTEGRITY MONITORING TIME
FIGURE B-13: ANALOG-TO-DIGITAL CONVERTER (ADC) CONFIGURATION FOR OVERTEMPERATURE INDICATOR

FIGURE B-14: TIMER3 CONFIGURATION FOR OVERTEMPERATURE MONITORING INTERRUPT

FIGURE B-15: HARDWARE LIMIT TIMER (HLT) IN TIMER2 CONFIGURATION FOR STALL DETECTION
After configuring all the peripherals, click the "Generate Code" button in the top left corner of the center panel. This will generate a `main.c` file to the project automatically. It will also initialize the module and leave an empty `while (1)` loop for custom code entry.
APPENDIX C: SOURCE CODE LISTING

The latest software version can be downloaded from the Microchip web site (www.microchip.com). The user will find the source code appended to the electronic version of this application note. The latest version is v3.0.
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, dsPIC, FlashFlex, fPWR, JukeBox, KEELoo, KEELoo logo, Kleer, LANCheck, MediaLB, MOST, MOST logo, MPLAB, OptoLzer, PIC, PICSTART, PIC<sup>®</sup> logo, RightTouch, SpyNIC, SST, SST Logo, SuperFlash and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

The Embedded Control Solutions Company and mTouch are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, BodyCom, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, ECAN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, KleerNet, KleerNet logo, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscent Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, RightTouch logo, REAL ICE, SOI, Serial Quad I/O, Total Endurance, TSHARC, USBCheck, VanSense, ViewSpan, WiperLock, Wireless DNA, 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.

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.

© 2016, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-0185-8
### 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  

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

**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  

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

**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  

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

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

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

**Canada - Toronto**  
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: 852-2943-5100  
Fax: 852-2401-3431  

**China - Beijing**  
Tel: 86-10-8569-7000  
Fax: 86-10-8528-2104  

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

**China - Chongqing**  
Tel: 86-23-8960-9588  
Fax: 86-23-8980-9500  

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

**China - Hangzhou**  
Tel: 86-571-8792-8115  
Fax: 86-571-8792-8116  

**China - Hong Kong SAR**  
Tel: 852-2943-5100  
Fax: 852-2401-3431  

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

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

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

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

**China - Shenzhen**  
Tel: 86-755-8864-2200  
Fax: 86-755-8203-1760  

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

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

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

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

**India - Pune**  
Tel: 91-20-3019-1500  

**Japan - Osaka**  
Tel: 81-6-6152-7160  
Fax: 81-6-6152-9310  

**Japan - Tokyo**  
Tel: 81-3-6880-3770  
Fax: 81-3-6880-3771  

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

**Korea - Seoul**  
Tel: 82-2-558-5932 or 82-2-558-5934  

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

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

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

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

**Taiwan - Hsin Chu**  
Tel: 886-3-5778-3466  
Fax: 886-3-5770-955  

**Taiwan - Kaohsiung**  
Tel: 886-7-213-7828  

**Taiwan - Taipei**  
Tel: 886-2-2508-8600  
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 - Dusseldorf**  
Tel: 49-2129-3766400  

**Germany - Karlsruhe**  
Tel: 49-721-625370  

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

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

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

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

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

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

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

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

07/14/15