INTRODUCTION

This application note discusses how to use the Enhanced, Capture, Compare, PWM (ECCP) on the PIC16F684 for bidirectional, brushed DC (BDC) motor control. Low-cost brushed DC motor control can be used in applications such as intelligent toys, small appliances and power tools. The PIC16F684 takes Microchip's Mid-Range Family of products to the next level with its new ECCP peripheral. The ECCP peripheral builds on the technology of the CCP module with added features such as four PWM channels for easy bidirectional motor control through the hardware. This application note focuses on using the ECCP in PWM mode using the full-bridge configuration. Using the ECCP allows easy interfacing to a full-bridge configuration for bidirectional BDC motor control.

This application note will discuss the following:
- Calculating ECCP PWM Parameters
- Initializing the ECCP in full-bridge PWM mode
- Bidirectional BDC Motor Control
- Sensorless Motor Control Feedback
- Example Application

Note: All equations referenced in this application note can be found in Appendix A.

CALCULATING ECCP PWM PARAMETERS

When working with the ECCP in PWM mode, the PWM frequency, duty cycle and resolution, there are three useful pieces of information to be calculated.

Frequency

Selecting a PWM frequency for the motor control application will effect the sound of the motor and the power transistor's switching speed. The human ear can detect frequencies ranging from 20 Hz-20 kHz. Generally, frequencies greater than 4 kHz are not audible to the human ear. Choosing a PWM frequency greater than 4 kHz helps reduce the humming sound heard while the motor is running.

The PWM period and frequency can be calculated using Equations 1 and 2 located in Appendix A.

Duty Cycle

Changing the PWM duty cycle will change the average voltage across the motor, which changes the motor's speed. The PWM duty cycle is calculated by using Equation 3. The average voltage across the BDC motor is calculated by using Equation 4.

Resolution

The PWM duty cycle resolution determines the amount of precision with which the duty cycle can be changed. For example, a 10-bit resolution allows 1024 possible values for the duty cycle where an 8-bit resolution only allows 256 values. The PWM frequency, PIC16F684 oscillator frequency and Timer2 prescaler all effect the resolution value. The maximum resolution is 10 bits. The PWM duty cycle resolution is calculated by using Equation 5.

INITIALIZING THE ECCP IN FULL-BRIDGE PWM MODE

When initializing the ECCP in full-bridge PWM mode, four registers need to be initialized.

PR2

The PR2 register effects the PWM frequency/period. The value to use for the PR2 register is calculated from Equation 6.
**CCPR1L:CCP1CON<5:4>**

The PWM duty cycle has a full resolution of 10 bits. Since all registers on the PIC16F684 are 8-bits wide, the 10 bits are spread over two registers. CCPR1L contains the upper 8 bits and CCP1CON<5:4> contains the lower 2 bits. The 10-bit value for CCPR1L:CCP1CON<5:4> is calculated by using Equation 7.

**CCP1CON**

In addition to storing the lower 2 bits of the 10-bit PWM duty cycle, CCP1CON is used to set up the ECCP in PWM mode using bits CCP1CON<3:0> and can change the motor direction using bits CCP1CON<7:6>. When setting up the ECCP in PWM mode, there are four possible configurations. These configurations accommodate H-bridges with MOSFETS that are active high, active low or a combination of both active high and active low. Motor direction can be changed in hardware by configuring bits CCP1CON<7:6> to be '01' for forward or '11' for reverse. The PIC16F684 ECCP hardware takes care of switching channels for activating and modulating the appropriate MOSFET drivers in the H-bridge.

**T2CON**

The T2CON register is used for setting up the Timer2 prescaler and turning on Timer2. The Timer2 prescaler is contained in bits T2CON<1:0> and is used in determining the PWM frequency, duty cycle and resolution. Timer2 must be turned on by setting bit T2CON<2> before the PWM signal will start. For an algorithm that calculates the Timer2 prescaler and PR2 values given a known PWM frequency (see Figure B-1 in Appendix B).
BIDIRECTIONAL BDC MOTOR CONTROL

The ECCP makes changing motor direction easy by configuring CCP1CON<7:6> to be '01' for forward (Figure 1) or '11' for reverse (Figure 2).

**FIGURE 1: FULL-BRIDGE FORWARD CURRENT FLOW DIAGRAM**

![Full-Bridge Forward Flow Diagram]

**FIGURE 2: FULL-BRIDGE REVERSE CURRENT FLOW DIAGRAM**

![Full-Bridge Reverse Flow Diagram]
LOW COST SENSORLESS MOTOR CONTROL FEEDBACK

Sensorless RPM Measurement

Low-cost RPM measurement can be performed with a BDC motor by measuring the back EMF voltage from the motor (see Figure 3). The BDC RPM is directly proportional to the back EMF voltage. Since a BDC motor can be modeled as an inductive load, the voltage across the motor is equivalent to the inductance multiplied by dI/dt. In this application, a 12V, 9600 max RPM BDC motor was used. To measure the back EMF voltage, turn the modulated FET “off.” This will cause the current to flow in the opposite direction. After initially shutting off the FET, dI/dt must stabilize before taking the measurement. In order to use the PICmicro® microcontroller A/D converter, the measured voltage must be between 0V and VDD. Since the back EMF voltage can be between 0V-12V, a voltage divider circuit is used to scale the back EMF voltage between 0V and VDD. Using Microchip’s MSP6S26 Programmable Gain Amplifier (PGA), a gain of 1 is used for buffering the scaled voltage that is being measured by the PIC16F684 A/D channel (see Equation 8 for calculating RPM).

Sensorless Current Measurement

Low-cost current measurements can be performed by using a current sensing resistor between the MOSFETS and ground (see Figure 4). To select a value appropriate for the resistance, consider the maximum amount of current allowed to flow through the resistor and the maximum amount of power dissipation.

In this application, a 0.1 ohm, 1W current sensing resistor was used with a maximum current of 3A. When 3A are flowing through the resistor, the ideal power dissipated in the resistor is 0.9W (see Equation 9). Also, when 3A are flowing through the resistor, the voltage across the resistor is 0.3V (see Equation 10). In order to get the most resolution from the 10-bit A/D converter, the voltage across the resistor at 3A must be amplified as close as possible to the PIC16F684 VDD, which is 5V in this application. Using Microchip’s MSP6S26 PGA, a gain of 16 will ideally give 4.8V, at the maximum 3A specified current (see Equation 11). A gain of 16 gives a 9.94 bit A/D resolution for measuring current (see Equations 12 and 13). The current through the resistor can then be computed using Equations 14, 15 and 16.

Since a PWM signal is used to drive the BDC motor, the H-bridge circuit only draws current during the high pulse-width of the PWM period. To obtain a current measurement, the voltage across the current sensing resistor is sampled over a PWM period. A sampling and averaging algorithm of taking measurements over multiple PWM periods is shown in Figure B-2 in Appendix B.
EXAMPLE APPLICATION

This example application demonstrates a low-cost BDC Motor Control system using the ECCP configured in full-bridge PWM mode (see Figure 5). The user interface allows the user to easily configure a BDC motor with the PIC16F684, adjust the PWM frequency and duty cycle, change the PIC16F684 internal oscillator frequency in real-time, and view RPM and current measurements. This application source code was written using the HI-TECH C Compiler, MPLAB® IDE, and the Microsoft Visual C++® 6.0 development platform.

FIGURE 5: MECHATRONICS BLOCK DIAGRAM
Firmware

The example firmware is responsible for many operations.

- Initializing the PIC16F684
- Sending bit-banged SPI™ commands to the PGA
- Receiving commands from the PC
- Modifying the PWM frequency and duty cycle
- Changing the motors direction
- Changing the internal oscillator frequency
- Taking A/D converter measurements for RPM and current

The PIC16F684 firmware implements a bit-banged RS-232 USART running at 9600 bps. See Appendix C for the RS-232 serial protocol used in this application note. The C source code can be downloaded from www.microchip.com. See Figure B-3 in Appendix B for the main program flow.

Software

The Windows® user interface provides the user a friendly environment for interfacing the BDC motor. The user interface allows the user to adjust the PWM frequency, duty cycle, motor direction and internal oscillator frequency. The user interface also displays the PWM frequency, duty cycle, resolution, RPM and current. The PC software is the host and sends commands to the PIC16F684 using RS-232. The Windows user interface source code can also be downloaded from www.microchip.com. The Windows user interface example is shown in Figure 6.

FIGURE 6: WINDOWS USER INTERFACE SCREEN

Hardware

The hardware used in this application note contains three major sections; a power stage for motor control, communication for RS-232, and measurement for RPM and current.

The power stage consists of a full H-bridge used for bidirectional BDC motor control. The PIC16F684 uses RC2-RC5 as the four ECCP pins that interface with the full H-bridge circuit.

The communication section consists of a RS-232 serial communication configuration. The PIC16F684 uses RA5 for sending and receiving RS-232 data.

The measurement section consists of Microchip's MSC6S26 multi-channel PGA and a voltage divider circuit for scaling the back EMF voltage, as discussed in the "Sensorless RPM Measurement" section.

The PIC16F684 communicates to the PGA via a 3-wire bit-banged SPI interface. The CS pin is connected to RA1. The SCK pin is connected to RA2. The SI pin is connected to RC0. The VREF pin is connected to GND. The RA0 pin is used as an analog input for measuring RPM and current. The RA0 pin is connected to the VOL pin on the PGA. Channel 0 on the PGA is used for RPM measurements. Channel 1 on the PGA is used for current measurements. See Figure D-1 in Appendix D for the schematic diagram of the hardware.
CONCLUSION

The PIC16F684 is well suited for low-cost bidirectional BDC motor control. This application note demonstrates how easy it is to calculate the necessary parameters for using the ECCP in PWM mode, initialize the necessary ECCP registers, use the ECCP for bidirectional BDC motor control, and implement sensorless RPM and current measurements. This application note concludes by showing a full application implementation using the PC Windows software, PIC16F684 firmware and Motor Control hardware.

REFERENCES

APPENDIX A: EQUATIONS

EQUATION 1: PWM FREQUENCY (HZ)

\[
Frequency = \frac{1}{Period}
\]

EQUATION 2: PWM PERIOD (SECONDS)

\[
Period = [(PR2 + 1)] \times 4 \times Tosc \times TMR2\text{Prescaler}
\]

EQUATION 3: DUTY CYCLE (SECONDS)

\[
DC = CCPR1L:CCP1CON<5:4> \times Tosc \times TMR2\text{Prescaler}
\]

EQUATION 4: VOLTAGE ACROSS BDC MOTOR (VOLTS)

\[
V_{BDC} = V_{DD} \times \left( \frac{DC}{Period} \right)
\]

EQUATION 5: RESOLUTION (BITS)

\[
\text{Resolution} = \frac{\log\left( \frac{F_{OSC}}{(FPWM \times TMR2\text{Prescaler})} \right)}{\log(2)}
\]

EQUATION 6: PR2

\[
PR2 = \left( \frac{Period}{4 \times Tosc \times TMR2\text{Prescaler}} \right) - 1
\]
EQUATION 7: \[ CCPR1L:CCP1CON<5:4> \]

\[
CCPR1L:CCP1CON<5:4> = \frac{DC}{Tos \times TMR2\text{Prescaler}}
\]

EQUATION 8: \[ RPM \]

\[
RPM = \left(1 - \left(\frac{ADRESH:ADRESL}{1024}\right)\right) \times RPM_{\text{MAX}}
\]

EQUATION 9: \[ POWER (W) \]

\[
P = I_{\text{MAX}}^2 \times R = 3^2 \times 0.1 = 0.9W
\]

EQUATION 10: \[ MAXIMUM VOLTAGE ACROSS RESISTOR (VOLTS) \]

\[
V_{\text{NominalMAX}} = I_{\text{MAX}} \times R = 3 \times 0.1 = 0.3V
\]

EQUATION 11: \[ MAXIMUM VOLTAGE AFTER AMPLIFICATION (VOLTS) \]

\[
V_{\text{GainMAX}} = V_{\text{NominalMAX}} \times Gain = 0.3 \times 16 = 4.8V
\]

EQUATION 12: \[ BITS OF RESOLUTION \]

\[
2^X = \frac{V_{\text{GainMAX}}}{V_{\text{DD}}} \times 1024, \text{ where } X \text{ is bits of resolution}
\]

EQUATION 13: \[ BITS OF RESOLUTION SOLVED FOR X \]

\[
X = \frac{\log\left(\frac{V_{\text{GainMAX}}}{V_{\text{DD}}} \times 1024\right)}{\log(2)} = \frac{\log\left(\frac{4.8}{5.0} \times 1024\right)}{\log(2)} = 9.94 \text{ bits}
\]
EQUATION 14: GAIN VOLTAGE MEASURED (VOLTS)

\[ V_{GAIN} = \left( \frac{ADRESH:ADRESL}{2^x} \right) \times V_{GAINMAX} \]

EQUATION 15: ACTUAL VOLTAGE ACROSS RESISTOR (VOLTS)

\[ V_{ACTUAL} = \frac{V_{GAIN}}{Gain} \]

EQUATION 16: CURRENT THROUGH RESISTOR (VOLTS)

\[ I = \frac{V_{ACTUAL}}{R} \]
FIGURE B-1: CALCULATING TIMER2 PRESCALER AND PR2 ALGORITHM GIVEN A PWM FREQUENCY

PR2 = unsigned int.
Prescaler = unsigned char.
i = unsigned char.
FIGURE B-2: PWM SAMPLING AND AVERAGING ALGORITHM

- Start
  - Measure PWM Period
  - Synchronize on PWM high edge
  - Delay
  - Start A/D Conversion
  - Is A/D Conversion complete?
    - No
    - Increment Delay
  - All samples taken?
    - Yes
    - Average samples
    - Log sample
    - Done
  - No

FIGURE B-3: MAIN ROUTINE

- Start
  - Initialize PIC16F684
  - Is command received?
    - No
    - Is command valid?
      - No
      - Send response
      - Process command
    - Yes
  - Yes
APPENDIX C: RS-232 SERIAL COMMUNICATIONS PROTOCOL

Since one-wire communication is being implemented, the command sent from the PC to the PIC16F684 will be echoed back. An example of this can be seen on the firmware version box in the Windows GUI. The firmware version box contains (f)[F1.0]. The PC command sent is (f). The PIC16F684 firmware response is [F1.0]. The general form of the command and response are described below as well as the commands implemented in the example application.

C.1 General Form

PC Command:
<command start><command><data> <command end>
Ex: (f).

PIC16F684 Response:
<response start><response><data><response end>
Ex: [F1.0]

Note 1: The <command> is lower case.
2: The <response> is the upper case of the <command>.
3: If there is no <data> to be sent, the <command end> can be the next character sent.
4: All <data> is sent in Hex format.
5: All <data> is sent Most Significant Byte first.
6: Invalid commands are ignored and responded with a [?].
7: Invalid <command start> is ignored and not responded to.
8: Commands and responses are currently set to 10 characters each, this can be adjusted in the source code on both the Windows software and PIC16F684 firmware.

C.2 Example Application Command Set

PR2 Command: Loads data into the PR2 register.
   PC Command: (aAF)
   PIC16F684 Response: [A]

CCPR1L Command: Loads data into the CCPR1L register.
   PC Command: (b1F)
   PIC16F684 Response: [B]

CCP1CON<5:4> Command: Loads data into CCP1CON<5:4>.
   PC Command: (c3)
   PIC16F684 Response: [C]

Timer2 Prescaler Command: Loads data into T2CON<1:0>.
   PC Command: (d0)
   PIC16F684 Response: [D]

   PC Command: (e6)
   PIC16F684 Response: [E]

FW Command: Requests the PIC16F684 firmware version.
   PC Command: (f)
   PIC16F684 Response: [F1.0]

Motor Control Command: Loads data into CCP1CON<7:6>.
   PC Command: (g3)
   PIC16F684 Response: [G]

RPM Measurement Command: Requests a RPM measurement.
   PC Command: (h)
   PIC16F684 Response: [H3FF]

Current Measurement Command: Requests a Current measurement.
   PC Command: (i)
   PIC16F684 Response: [I2BC]
FIGURE D-2: BDC MOTOR CONTROL SCHEMATIC
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 intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip’s products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks

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

AmpLab, FilterLab, microID, MXDEV, MXLAB, PICMASTER, SEEVAL, SmartShunt and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Application Maestro, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, PICkit, PICDEM, PICDEM.net, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, rLAB, rPIC, Select Mode, SmartSensor, SmartTel and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

Serialized Quick Turn Programming (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.

© 2003, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.