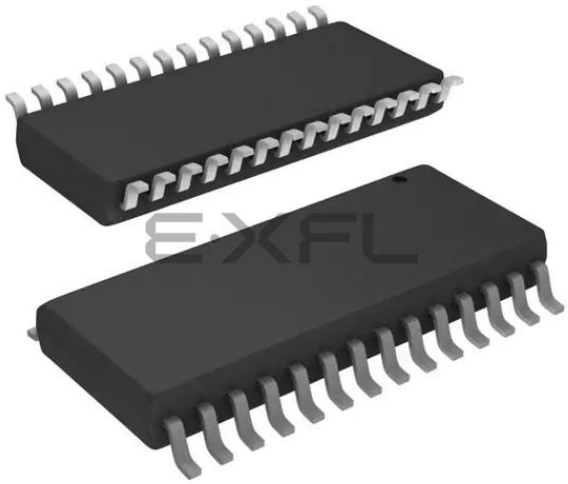


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	4MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	22
Program Memory Size	7KB (4K x 14)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	192 x 8
Voltage - Supply (Vcc/Vdd)	4V ~ 5.5V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16c63a-04e-so

TIP #3 Read Three States From One Pin

To check state Z:

- Drive output pin high
- Set to Input
- Read 1
- Drive output pin low
- Set to Input
- Read 0

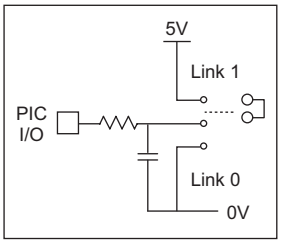
To check state 0:

- Read 0 on pin

To check state 1:

- Read 1 on pin

Figure 3-1



State	Link 0	Link 1
0	closed	open
1	open	closed
NC	open	open

Jumper has three possible states: not connected, Link 1 and Link 0. The capacitor will charge and discharge depending on the I/O output voltage allowing the "not connected" state. Software should check the "not connected" state first by driving I/O high, reading 1 and driving I/O low and reading 0. The "Link 1" and "Link 0" states are read directly.

TIP #4 Reading DIP Switches

The input of a timer can be used to test which switch(s) is closed. The input of Timer1 is held high with a pull-up resistor. Sequentially, each switch I/O is set to input and Timer1 is checked for an increment indicating the switch is closed.

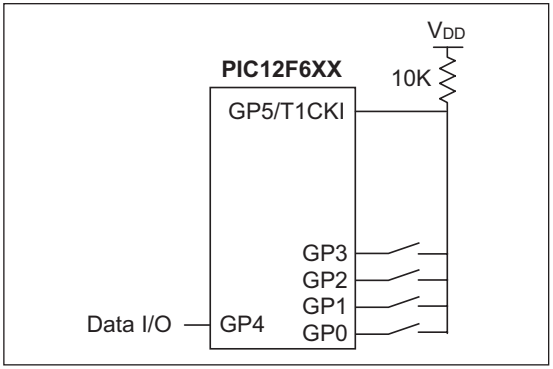
Example 4-1

```
        movlw   b'11111111'
        movwf   TRISIO
        movwf   DIP
        movlw   b'00000111'
        movwf   T1CON
        movlw   b'11111110'
        movwf   Mask
        clrf    GPIO
LOOP
        clrf    TMR1L
        movf    Mask,W
        movwf   TRISIO
        btfsc   TMR1L,0
        andwf   DIP,F
        bsf     STATUS,C
        rlf     Mask,F
        btfsc   Mask,4
        goto    Loop
        retlw   0
```

Each bit in the DP register represents its corresponding switch position. By setting Timer1 to FFFFh and enabling its interrupt, an increment will cause a rollover and generate an interrupt. This will simplify the software by eliminating the bit test on the TMR1L register.

Sequentially set each GPIO to an input and test for TMR1 increment (or 0 if standard I/O pin is used).

Figure 4-1



TIPS ‘N TRICKS WITH SOFTWARE

To reduce costs, designers need to make the most of the available program memory in MCUs. Program memory is typically a large portion of the MCU cost. Optimizing the code helps to avoid buying more memory than needed. Here are some ideas that can help reduce code size.

TIP #15 Delay Techniques

- Use GOTO “next instruction” instead of two NOPs.
- Use CALL Rtrn as quad, 1 instruction NOP (where “Rtrn” is the exit label from existing subroutine).

Example 15-1

```
        NOP
        NOP             ;2 instructions, 2 cycles

        GOTO    $+1     ;1 instruction, 2 cycles

        CALL Rtrn       ;1 instruction, 4 cycles
        . . .
Rtrn    RETURN
```

MCUs are commonly used to interface with the “outside world” by means of a data bus, LEDs, buttons, latches, etc. Because the MCU runs at a fixed frequency, it will often need delay routines to meet setup/hold times of other devices, pause for a handshake or decrease the data rate for a shared bus.

Longer delays are well-suited for the DECFSZ and INCFSZ instructions where a variable is decremented or incremented until it reaches zero when a conditional jump is executed. For shorter delays of a few cycles, here a few ideas to decrease code size.

For a two-cycle delay, it is common to use two NOP instructions which uses two program memory locations. The same result can be achieved by using “goto $+1”. The “$” represents the current program counter value in MPASM™ Assembler. When this instruction is encountered, the MCU will jump to the next memory location. This is what it would have done if two NOP’s were used but since the GOTO instruction uses two instruction cycles to execute, a two-cycle delay was created. This created a two-cycle delay using only one location of program memory.

To create a four-cycle delay, add a label to an existing RETURN instruction in the code. In this example, the label “Rtrn” was added to the RETURN of subroutine that already existed somewhere in the code. When executing “CALL Rtrn”, the MCU delays two instruction cycles to execute the CALL and two more to execute the RETURN. Instead of using four NOP instructions to create a four-cycle delay, the same result was achieved by adding a single CALL instruction.

Static Power Reduction Tips n' Tricks

The following tips and tricks will help reduce the power consumption of a device while it is asleep. These tips allow an application to stay asleep longer and to consume less current while sleeping.

TIP #16 Deep Sleep Mode

In Deep Sleep mode, the CPU and all peripherals except RTCC, DSWDT and LCD (on LCD devices) are not powered. Additionally, Deep Sleep powers down the Flash, SRAM, and voltage supervisory circuits. This allows Deep Sleep mode to have lower power consumption than any other operating mode. Typical Deep Sleep current is less than 50 nA on most devices. Four bytes of data are retained in the DSGPRx registers that can be used to save some critical data required for the application. While in Deep Sleep mode, the states of I/O pins and 32 kHz crystal oscillator (Timer1/SOSC) are maintained so that Deep Sleep mode does not interrupt the operation of the application. The RTCC interrupt, Ultra Low Power Wake-up, DSWDT time-out, External Interrupt 0 (INT0), MCLR or POR can wake-up the device from Deep Sleep. Upon wake-up the device resumes operation at the reset vector.

Deep Sleep allows for the lowest possible static power in a device. The trade-off is that the firmware must re-initialize after wake-up. Therefore, Deep Sleep is best used in applications that require long battery life and have long sleep times. Refer to the device datasheets and Family Reference Manuals for more information on Deep Sleep and how it is used.

TIP #17 Extended WDT and Deep Sleep WDT

A commonly used source to wake-up from Sleep or Deep Sleep is the Watchdog Timer (WDT) or Deep Sleep Watchdog Timer (DSWDT). The longer the PIC MCU stays in Sleep or Deep Sleep, the less power consumed. Therefore, it is appropriate to use as long a timeout period for the WDT as the application will allow.

The WDT runs in all modes except for Deep Sleep. In Deep Sleep, the DSWDT is used instead. The DSWDT uses less current and has a longer timeout period than the WDT. The timeout period for the WDT varies by device, but typically can vary from a few milliseconds to up to 2 minutes. The DSWDT time-out period can be programmed from 2.1ms to 25.7days

TIP #18 Low Power Timer1 Oscillator and RTCC

nanoWatt XLP microcontrollers all have a robust Timer1 oscillator (SOSC on PIC24) which draws less than 800 nA. nanoWatt technology devices offer a low power Timer1 oscillator which draws 2-3 uA. Some devices offer a selectable oscillator which can be used in either a low-power or high-drive strength mode to suit both low power or higher noise applications. The Timer1 counter and oscillator can be used to generate interrupts for periodic wakes from Sleep and other power managed modes, and can be used as the basis for a real-time clock. Timer1/SOSC wake-up options vary by device. Many nanoWatt XLP devices have a built-in hardware Real-Time Clock and Calendar (RTCC), which can be configured for wake-up periods from 1 second to many years.

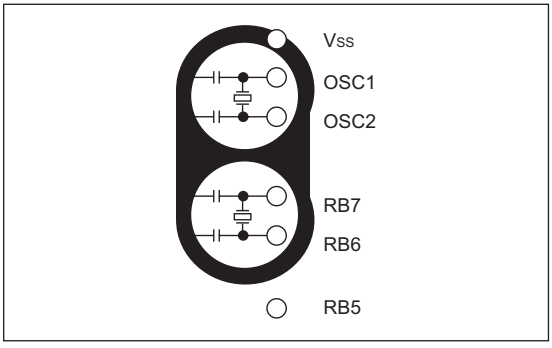
Some nanoWatt devices and all nanoWatt XLP devices can also use the Timer1/SOSC oscillator as the system clock source in place of the main oscillator on the OSC1/OSC2 pins. By reducing execution speed, total current consumption can be reduced.

TIP #19 Low Power Timer1 Oscillator Layout

Applications requiring very low power Timer1/SOSC oscillators on nanoWatt and nanoWatt XLP devices must take PCB layout into consideration. The very low power Timer1/SOSC oscillators on nanoWatt and nanoWatt XLP devices consume very little current, and this sometimes makes the oscillator circuit sensitive to neighboring circuits. The oscillator circuit (crystal and capacitors) should be located as close as possible to the microcontroller.

No circuits should be passing through the oscillator circuit boundaries. If it is unavoidable to have high-speed circuits near the oscillator circuit, a guard ring should be placed around the oscillator circuit and microcontroller pins similar to the figure below. Placing a ground plane under the oscillator components also helps to prevent interaction with high speed circuits.

Figure 19-1: Guard Ring Around Oscillator Circuit and MCU Pins



TIP #20 Use LVD to Detect Low Battery

The Low Voltage Detect (LVD) interrupt present in many PIC MCUs is critical in battery based systems. It is necessary for two reasons. First, many devices cannot run full speed at the minimum operating voltage. In this case, the LVD interrupt indicates when the battery voltage is dropping so that the CPU clock can be slowed down to an appropriate speed, preventing code misexecution. Second, it allows the MCU to detect when the battery is nearing the end of its life, so that a low battery indication can be provided and a lower power state can be entered to maximize battery lifetime. The LVD allows these functions to be implemented without requiring the use of extra analog channels to measure the battery level.

TIP #21 Use Peripheral FIFO and DMA

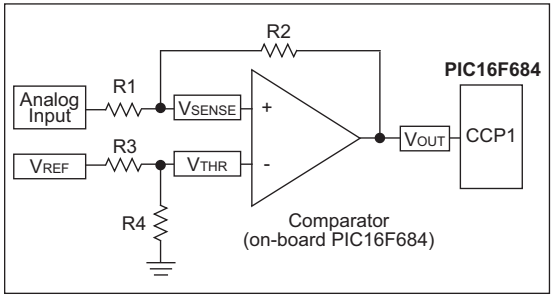
Some devices have peripherals with DMA or FIFO buffers. These features are not just useful to improve performance; they can also be used to reduce power. Peripherals with just one buffer register require the CPU to stay operating in order to read from the buffer so it doesn’t overflow. However, with a FIFO or DMA, the CPU can go to sleep or idle until the FIFO fills or DMA transfer completes. This allows the device to consume a lot less average current over the life of the application.

TIP #6 Measuring the Period of an Analog Signal

Microcontrollers with on-board Analog Comparator module(s), in addition to a CCP (or ECCP) module, can easily be configured to measure the period of an analog signal.

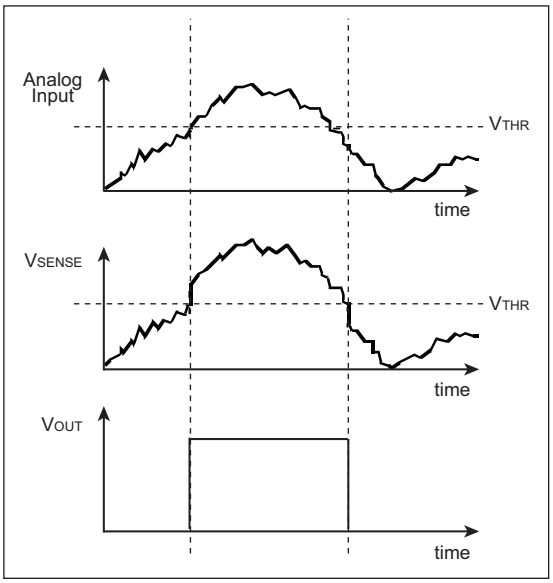
Figure 6-1 shows an example circuit using the peripherals of the PIC16F684.

Figure 6-1: Circuit



R3 and R4 set the threshold voltage for the comparator. When the analog input reaches the threshold voltage, VOUT will toggle from low to high. R1 and R2 provide hysteresis to insure that small changes in the analog input won't cause jitter in the circuit. Figure 6-2 demonstrates the effect of hysteresis on the input. Look specifically at what VSENSE does when the analog input reaches the threshold voltage.

Figure 6-2: Signal Comparison



The CCP module, configured in Capture mode, can time the length between the rising edges of the comparator output (VOUT.) This is the period of the analog input, provided the analog signal reaches VTHR during every period.

PWM TIPS ‘N TRICKS

The ECCP and CCP modules produce a 10-bit resolution Pulse-Width Modulated (PWM) waveform on the CCPx pin. The ECCP module is capable of transmitting a PWM signal on one of four pins, designated P1A through P1D. The PWM modes available on the ECCP module are:

- Single output (P1A only)
- Half-bridge output (P1A and P1B only)
- Full-bridge output forward
- Full-bridge output reverse

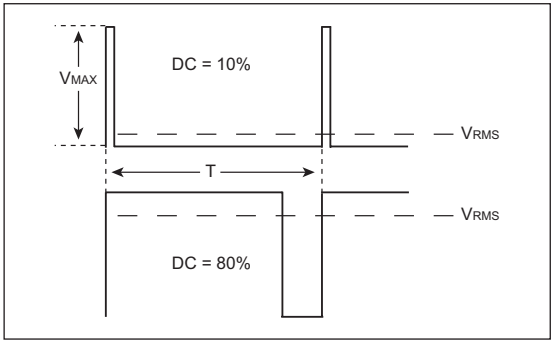
One of the following configurations must be chosen when using the ECCP module in PWM Full-bridge mode:

- P1A, P1C active-high; P1B, P1D active-high
- P1A, P1C active-high; P1B, P1D active-low
- P1A, P1C active-low; P1B, P1D active-high
- P1A, P1C active-low; P1B, P1D active-low

“Why Would I Use PWM Mode?”

As the next set of Tips ‘n Tricks demonstrate, Pulse-Width Modulation (PWM) can be used to accomplish a variety of tasks from dimming LEDs to controlling the speed of a brushed DC electric motor. All these applications are based on one basic principle of PWM signals – as the duty cycle of a PWM signal increases, the average voltage and power provided by the PWM increases. Not only does it increase with duty cycle, but it increases linearly. The following figure illustrates this point more clearly. Notice that the RMS and maximum voltage are functions of the duty cycle (DC) in the following Figure 12-3.

Figure 12-3: Duty Cycle Relation to V_{RMS}



Equation 12-1 shows the relation between V_{RMS} and V_{MAX}.

Equation 12-1: Relation Between V_{RMS} and V_{MAX}

$$V_{RMS} = DC \times V_{MAX}$$

TIP #13 Deciding on PWM Frequency

In general, PWM frequency is application dependent although two general rules-of-thumb hold regarding frequency in all applications. They are:

1. As frequency increases, so does current requirement due to switching losses.
2. Capacitance and inductance of the load tend to limit the frequency response of a circuit.

In low-power applications, it is a good idea to use the minimum frequency possible to accomplish a task in order to limit switching losses. In circuits where capacitance and/or inductance are a factor, the PWM frequency should be chosen based on an analysis of the circuit.

Motor Control

PWM is used extensively in motor control due to the efficiency of switched drive systems as opposed to linear drives. An important consideration when choosing PWM frequency for a motor control application is the responsiveness of the motor to changes in PWM duty cycle. A motor will have a faster response to changes in duty cycle at higher frequencies. Another important consideration is the sound generated by the motor. Brushed DC motors will make an annoying whine when driven at frequencies within the audible frequency range (20 Hz-4 kHz.) In order to eliminate this whine, drive brushed DC motors at frequencies greater than 4 kHz. (Humans can hear frequencies at upwards of 20 kHz, however, the mechanics of the motor winding will typically attenuate motor whine above 4 kHz).

LED and Light Bulbs

PWM is also used in LED and light dimmer applications. Flicker may be noticeable with rates below 50 Hz. Therefore, it is generally a good rule to pulse-width modulate LEDs and light bulbs at 100 Hz or higher.

TIP #14 Unidirectional Brushed DC Motor Control Using CCP

Figure 14-1: Brushed DC (BDC) Motor Control Circuit

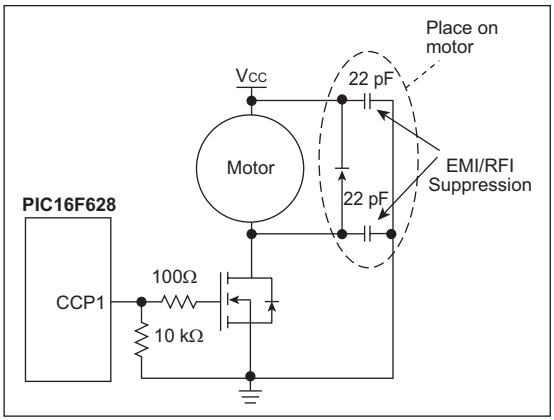


Figure 14-1 shows a unidirectional speed controller circuit for a brushed DC motor. Motor speed is proportional to the duty cycle of the PWM output on the CCP1 pin. The following steps show how to configure the PIC16F628 to generate a 20 kHz PWM with 50% duty cycle. The microcontroller is running on a 20 MHz crystal.

Step #1: Choose Timer2 Prescaler

a) F_{PWM} = Fosc/((PR2+1)*4*prescaler) = 19531 Hz for PR2 = 255 and prescaler of 1

b) This frequency is lower than 20 kHz, therefore a prescaler of 1 is adequate.

Step #2: Calculate PR2

PR2 = Fosc/(F_{PWM}*4*prescaler) – 1 = 249

Step #3: Determine CCPR1L and CCP1CON<5:4>

a) CCPR1L:CCP1CON<5:4> = DutyCycle*0x3FF = 0x1FF

b) CCPR1L = 0x1FF >> 2 = 0x7F, CCP1CON<5:4> = 3

Step #4: Configure CCP1CON

The CCP module is configured in PWM mode with the Least Significant bits of the duty cycle set, therefore, CCP1CON = '`b001111000`'.

CHAPTER 4
PIC® Microcontroller Comparator Tips ‘n Tricks

Table Of Contents

TIPS ‘N TRICKS INTRODUCTION

Microchip continues to provide innovative products that are smaller, faster, easier to use and more reliable. The Flash-based PIC® microcontrollers (MCUs) are used in a wide range of everyday products from smoke detectors to industrial, automotive and medical products.

The PIC12F/16F Family of devices with on-chip voltage comparators merge all the advantages of the PIC MCU architecture and the flexibility of Flash program memory with the mixed signal nature of a voltage comparator. Together they form a low-cost hybrid digital/analog building block with the power and flexibility to work in an analog world.

The flexibility of Flash and an excellent development tool suite, including a low-cost In-Circuit Debugger, In-Circuit Serial Programming™ (ICSP™) and MPLAB® ICE 2000 emulation, make these devices ideal for just about any embedded control application.

The following series of Tips ‘n Tricks can be applied to a variety of applications to help make the most of discrete voltage comparators or microcontrollers with on-chip voltage comparators.

TIP #11 PWM Generator

This tip shows how the multi-vibrator (ramp wave) can be used to generate a voltage controlled PWM signal. The ramp wave multi-vibrator operates as described in Tip #9, generating a positive going ramp wave. A second comparator compares the instantaneous voltage of the ramp wave with the incoming voltage to generate the PWM output (see Figure 11-2).

When the ramp starts, it is below the input voltage, and the output of the second comparator is pulled high starting the PWM pulse. The output remains high until the ramp wave voltage exceeds the input, then the output of the second comparator goes low ending the PWM pulse. The output of the second comparator remains low for the remainder of the ramp waveform. When the ramp waveform returns to zero at the start of the next cycle, the second comparator output goes high again and the cycle starts over.

Figure 11-1: PWM Wave Forms

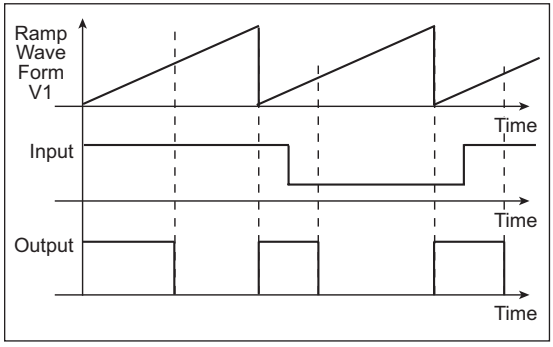
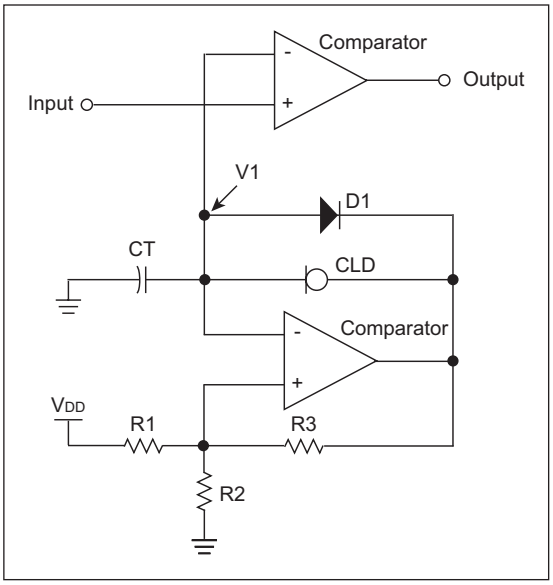


Figure 11-2: PWM Circuit



To design a PWM generator, start with the design of a ramp wave multi-vibrator using the design procedure from Tip #9. Choose high and low threshold voltages for the multi-vibrators hysteresis feedback that are slightly above and below the desired PWM control voltages.

Note: The PWM control voltage will produce a 0% duty cycle for inputs below the low threshold of the multi-vibrator. A control voltage greater than the high threshold voltage will produce a 100% duty cycle output.

Using the example values from Tip #9 will result in a minimum pulse width at an input voltage of 1.7V and a maximum at an input of 3.2V.

TIP #12 Making an Op Amp Out of a Comparator

When interfacing to a sensor, some gain is typically required to match the full range of the sensor to the full range of an ADC. Usually this is done with an operational amplifier, however, in cost sensitive applications, an additional active component may exceed the budget. This tip shows how an on-chip comparator can be used as an op amp like gain stage for slow sensor signals. Both an inverting and non-inverting topology are shown (see Figure 12-1 and Figure 12-2).

Figure 12-1: Non-Inverting Amplifier

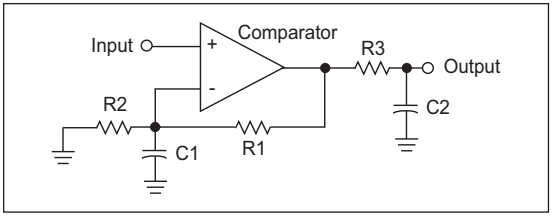
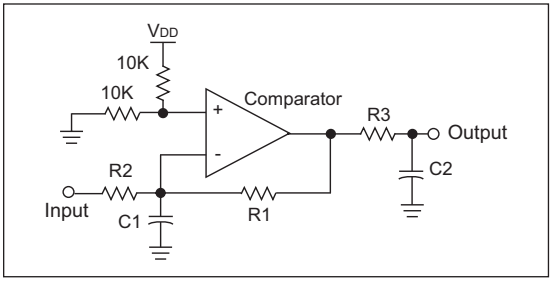


Figure 12-2: Inverting Amplifier



To design a non-inverting amplifier, choose resistors R1 and R2 using the Gain formula for an op amp non-inverting amplifier (see Equation 12-1).

Equation 12-1

$$\text{Gain} = \frac{R1 + R2}{R2}$$

Once the gain has been determined, values for R3 and C2 can be determined. R3 and C2 form a low-pass filter on the output of the amplifier. The corner frequency of the low pass should be 2 to 3 times the maximum frequency of the signal being amplified to prevent attenuation of the signal, and R3 should be kept small to minimize the output impedance of the amplifier. Equation 12-2 shows the relationship between R3, C2 and the corner frequency of the low pass filter.

Equation 12-2

$$F_{CORNER} = \frac{1}{2 * \ * R3 * C2}$$

A value for C1 can then be determined using Equation 12-3. The corner frequency should be the same as Equation 12-3.

Equation 12-3

$$F_{CORNER} = \frac{1}{2 * \ * (R1 \text{ II } R2) * C2}$$

To design an inverting amp, choose resistors R1 and R2 using the Gain formula for an op amp inverting amplifier (see Equation 12-4).

Equation 12-4

$$\text{Gain} = \frac{R1}{R2}$$

Then choose values for the resistor divider formed by R4 and R5. Finally choose C1 and C2 as shown in the non-inverting amplifier design.

Example:

- For C2 will set the corner F
- Gain = 6.156, R1 = R3 = 19.8k
- R2 = 3.84k, C1 = .047 μF, F_{CORNER} = 171 Hz
- C2 = .22 μF

CHAPTER 5
DC Motor Control Tips ‘n Tricks

Table Of Contents

TIPS ‘N TRICKS INTRODUCTION

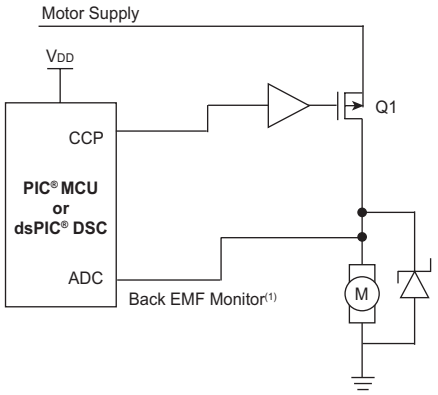
Every motor control circuit can be divided into the drive electronics and the controlling software. These two pieces can be fairly simple or extremely complicated depending upon the motor type, the system requirements and the hardware/software complexity trade-off. Generally, higher performance systems require more complicated hardware. This booklet describes many basic circuits and software building blocks commonly used to control motors. The booklet also provides references to Microchip application notes that describe many motor control concepts in more detail. The application notes can be found on the Microchip web site at www.microchip.com.

Additional motor control design information can be found at the Motor Control Design Center (www.microchip.com/motor).

TIP #7 Position/Speed Sensing

The motor RPM can be measured by understanding that a motor is a generator. As long as the motor is spinning, it will produce a voltage that is proportional to the motors RPM. This is called back EMF. If the PWM signal to the motor is turned off and the voltage across the windings is measured, the back EMF voltage can be sensed from there and the RPM’s can be known.

Figure 7-1: Back EMF Motor Speed Sensing



Note 1: If motor voltage is greater than V_{DD}, an attenuator will be required. Sample back EMF while Q1 is off.

Rotary Encoder Sensing

Rotary encoders are typically used to provide direct physical feedback of motor position, and/or speed. A rotary encoder consists of a rotary element attached to the motor that has a physical feature, measured by a stationary component. The measurements can yield motor speed and sometimes they can provide a motor position. Rotary encoders are built using many different technologies. The most common type is an optical rotary encoder. The optical rotary encoder is used in the computer mice that have a ball. It is built with an encoder disc that is attached to the motor. The encoder disc has many radial slots cut into the disc at a specific interval. An LED and a photo detector are used to count the slots as they go by. By timing the rate that the slots go by, the speed of rotation can be determined.

Sensing motor position requires a second LED and photo detector. The second sensor pair is mounted so the output pulses are 90° out of phase from the first pair. The two outputs represent the motion of the encoder disc as a quadrature modulated pulse train. By adding a third index signal, that pulses once for each revolution, the exact position of the rotor can be known.

An encoder with quadrature outputs can be used to track relative position from a known reference point. Another type of encoder uses a binary encoded disk so that the exact rotor position is always known. This type of encoder is called an absolute encoder.

TIP #7 Driving Common Backlights

Any application that operates in a low light condition requires a backlight. Most low-cost applications use one of the following backlights:

1) Electroluminescent (EL)

2) LEDs in series

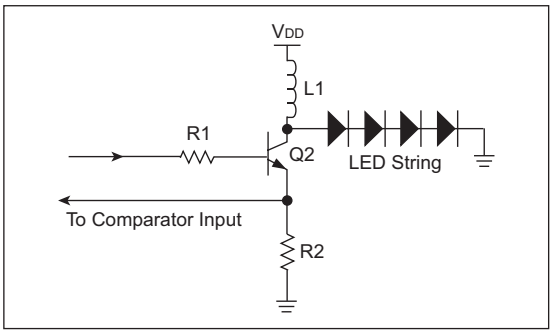
3) LEDs in parallel

Other backlight technologies, such as CCFL, are more commonly used in high brightness graphical panels, such as those found in laptop computers. The use of white LEDs is also more common in color LCDs, where a white light source is required to generate the colors.

Driving an EL panel simply requires an AC signal. You may be able to generate this signal simply by using an unused segment on the LCD controller. The signal can also be generated by a CCP module or through software. The AC signal will need to pass through a transformer for voltage gain to generate the required voltage across the panel.

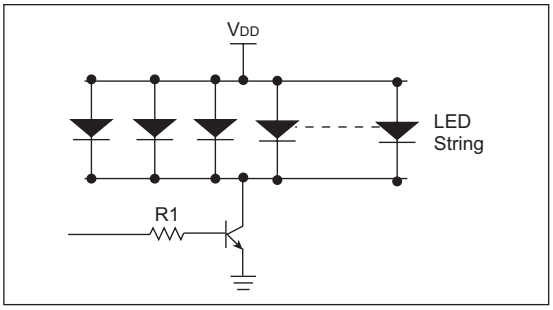
LEDs in series can be easily driven with a boost power supply. In the following diagram, a simple boost supply is shown. In this circuit, a pulse is applied to the transistor. The pulse duration is controlled by current through R2. When the pulse is turned off, the current stored in the inductor will be transferred to the LEDs. The voltage will rise to the level required to drive the current through the LEDs. The breakdown voltage of the transistor must be equal to the forward voltage of the LEDs multiplied by the number of LEDs. The comparator voltage reference can be adjusted in software to change the output level of the LEDs.

Figure 7-1: Simple Boost Supply



If the LEDs are in parallel, the drive is much simpler. In this case, a single transistor can be used to sink the current of many LEDs in parallel. The transistor can be modulated by PWM to achieve the desired output level. If V_{DD} is higher than the maximum forward voltage, a resistor can be added to control the current, or the transistor PWM duty cycle can be adjusted to assure the LEDs are operating within their specification.

Figure 7-2: LEDs in Parallel

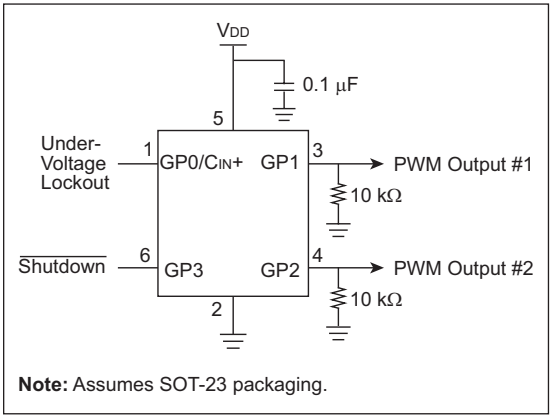


TIP #2 A Start-Up Sequencer

Some new devices have multiple voltage requirements (e.g., core voltages, I/O voltages, etc.). The sequence in which these voltages rise and fall may be important.

By expanding on the previous tip, a start-up sequencer can be created to control two output voltages. Two PWM outputs are generated to control the shutdown pins of two SMPS controllers. Again, this type of control only works on controllers that respond quickly to changes on the shutdown pin (such as those that do cycle-by-cycle limiting).

Figure 2-1: Multiple PWM Output Soft-Start Controller



Note: Assumes SOT-23 packaging.

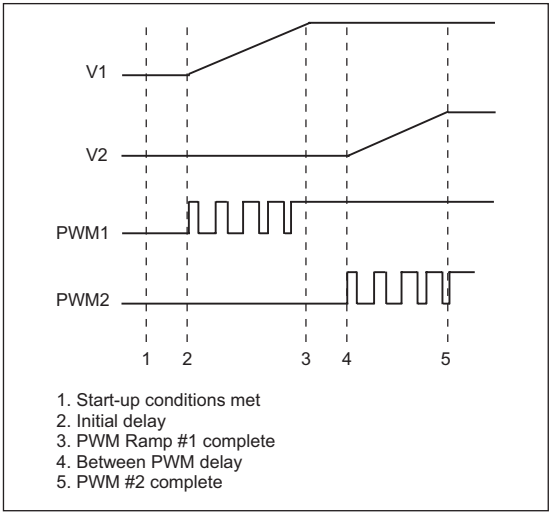
This design uses the PIC MCU comparator to implement an under-voltage lockout. The input on the GP0/CIN+ pin must be above the internal 0.6V reference for soft-start to begin, as shown in Figure 2-2.

Two conditions must be met in order for the soft-start sequence to begin:

1. The shutdown pin must be held at VDD (logic high).
2. The voltage on GP0 must be above 0.6V.

Once both start-up conditions are met, the sequences will delay and PWM #1 will ramp from 0% to 100%. A second delay allows the first voltage to stabilize before the sequencer ramps PWM #2 from 0% to 100%. All delays and ramp times are under software control and can be customized for specific applications. If either soft-start condition becomes invalid, the circuit will shutdown the SMPS controllers.

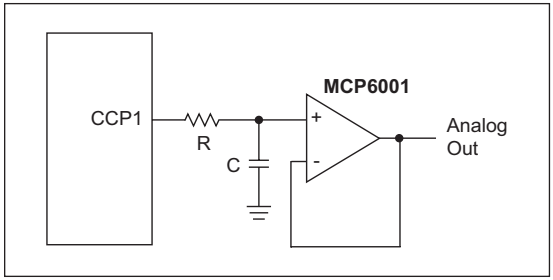
Figure 2-2: Timing Diagram



Example software is provided for the PIC10F200 which was taken from TB093, “*Multiple PWM Output Soft-Start Controller for Switching Power Supplies*” (DS91093).

TIP #11 Generating a Reference Voltage with a PWM Output

Figure 11-1: Low-Pass Filter



A PWM signal can be used to create a Digital-to-Analog Converter (DAC) with only a few external components. Conversion of PWM waveforms to analog signals involves the use of an analog low-pass filter. In order to eliminate unwanted harmonics caused by a PWM signal, the PWM frequency (F_{PWM}) should be significantly higher than the bandwidth (F_{BW}) of the desired analog signal. Equation 11-1 shows this relation.

Equation 11-1

$$F_{PWM} = K \bullet F_{BW}$$

Where harmonics decrease as K increases.

R and C are chosen based on the following equation:

Equation 11-2

$$RC = 1/(2 \bullet \pi \bullet F_{BW})$$

Where harmonics decrease as K increases.

Choose the R value based on drive capability and then calculate the required C value. The attenuation of the PWM frequency for a given RC filter is shown in Equation 11-3.

Equation 11-3

$$Att(dB) = -10 \bullet \log [1 + (2 \pi \bullet F_{PWM} \bullet RC)^2]$$

If the attenuation calculated in Equation 11-3 is not sufficient, then K must be increased in Equation 11-1.

In order to sufficiently attenuate the harmonics, it may be necessary to use small capacitor values or large resistor values. Any current draw will effect the voltage across the capacitor. Adding an op amp allows the analog voltage to be buffered and, because of this, any current drawn will be supplied by the op amp and not the filter capacitor.

For more information on using a PWM signal to generate an analog output, refer to AN538, “*Using PWM to Generate Analog Output*” (DS00538).

TIP #12 Using Auto-Shutdown CCP

PWM Auto-Shutdown

Several of Microchip’s PIC MCUs, such as the PIC16F684, PIC16F685 and PIC16F690, have a PWM auto-shutdown feature. When auto-shutdown is enabled, an event can terminate the current PWM pulse and prevent subsequent pulses unless the event is cleared. The ECCP can be setup to automatically start generating pulses again once the event clears.

Figure 12-1: PWM Auto-Shutdown Timing

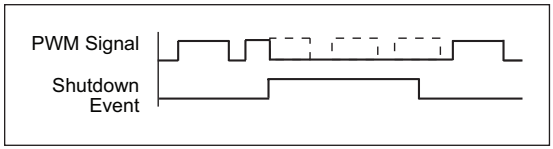
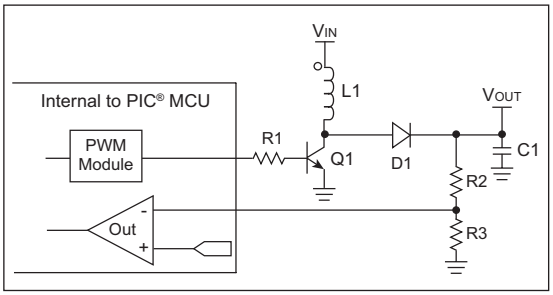


Figure 12-1 shows an example timing for the PWM auto-shutdown. When the shutdown event occurs, the current pulse is immediately terminated. In this example, the next two pulses are also terminated because the shutdown event had not been cleared by the beginning of the pulse period. After the event has cleared, pulses are allowed to resume, but only at the beginning of a pulse period.

Using Auto-Shutdown to Create a Boost Supply

By using the auto-shutdown feature, a very simple SMPS can be created. Figure 12-2 shows an example boost power supply.

Figure 12-2: Boost Power Supply



This power supply configuration has several unique features:

1. The switching frequency is determined by the PWM frequency and, therefore, can be changed at any time.
2. The maximum on-time is determined by the PWM duty cycle and, therefore, can be changed any time. This provides a very easy way to implement soft-start.
3. On PIC MCUs that have a programmable reference module, the output voltage can be configured and changed at any time.

The topology can also be re-arranged to create other types of power supplies.

Example software is provided for the PIC16F685 (but can be adapted to any PIC MCU with the ECCP module). The software configures the PWM and comparator modules as shown in Figure 12-2.

TIP #14 Brushless DC Fan Speed Control

There are several methods to control the speed of a DC brushless fan. The type of fan, allowable power consumption and the type of control desired are all factors in choosing the appropriate type.

Figure 14-1: Low-Side PWM Drive

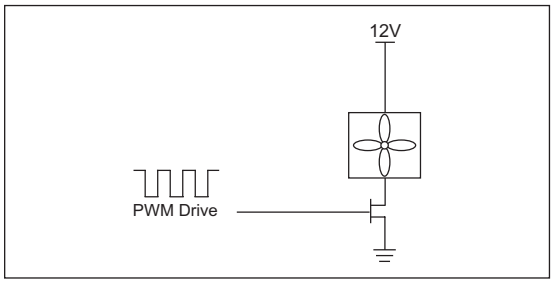
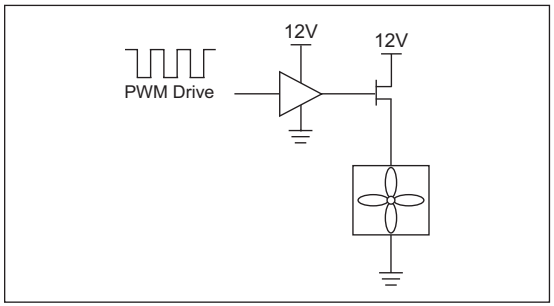


Figure 14-2: High-Side PWM Drive



Method 1 – Pulse-Width Modulation

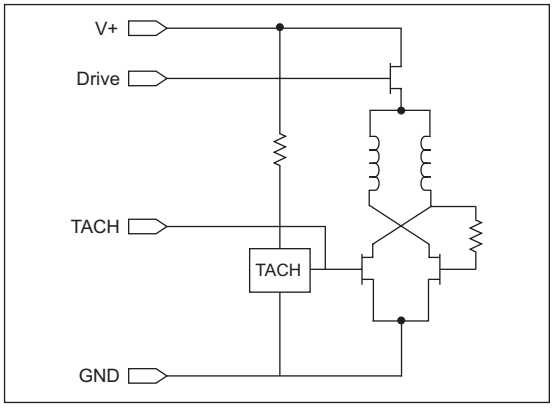
As shown in Figure 14-1 and Figure 14-2, a simple PWM drive may be used to switch a two-wire fan on and off. While it is possible to use the circuit in Figure 14-1 without a high-side MOSFET driver, some manufacturers state that switching on the low side of the fan will void the warranty.

Because of this, it is necessary to switch the high side of the fan in order to control the speed. The simplest type of speed control is ‘on’ or ‘off’. However, if a higher degree of control is desired, PWM can be used to vary the speed of the fan.

For 3-wire fans, the tachometer output will not be accurate if PWM is used. The sensor providing the tachometer output on 3-wire fans is powered from the same supply as the fan coils, thus using a PWM to control fan speed will render the fan’s tachometer inaccurate.

One solution is to use a 4-wire fan which includes both the tachometer output and a drive input. Figure 14-3 shows a diagram of a 4-wire fan.

Figure 14-3: Typical 4-Wire Fan



A 4-wire fan allows speed to be controlled using PWM via the Drive line. Since power to the tachometer sensor is not interrupted, it will continue to output the correct speed.

TIP #19 Execution-Indexed Software State Machine

Another common type of state machine is the execution-indexed state machine. This type of state machine uses a state variable in order to determine what is executed. In C, this can be thought of as the switch statement structure as shown in Example 19-1.

Example 19-1: Example Using Switch Statement

```
SWITCH (State)
{
  CASE 0: IF (in_key()==5)  THEN state = 1;
          Break;
  CASE 1: IF (in_key()==8)  THEN State = 2;
                            Else State = 0;
          Break;
  CASE 2: IF (in_key()==3)  THEN State = 3;
                            Else State = 0;
          Break;
  CASE 3: IF (in_key()==2)  THEN UNLOCK();
                            Else State = 0;
          Break;
}
```

Each time the software runs through the loop, the action taken by the state machine changes with the value in the state variable. By allowing the state machine to control its own state variable, it adds memory, or history, because the current state will be based on previous states. The microcontroller is able to make current decisions based on previous inputs and data.

In assembly, an execution-indexed state machine can be implemented using a jump table.

Example 19-2: Example Using a Jump Table

```
  MOVFW   state      ;load state into w
  ADDWF   PCL,f      ;jump to state
                     ;number
  GOTO    state0     ;state 0
  GOTO    state1     ;state 1
  GOTO    state2     ;state 2
  GOTO    state3     ;state 3
  GOTO    state4     ;state 4
  GOTO    state5     ;state 5
```

In Example 19-2, the program will jump to a GOTO statement based on the state variable. The GOTO statement will send the program to the proper branch. Caution must be taken to ensure that the variable will never be larger than intended. For example, six states (000 to 101) require a three-bit state variable. Should the state variable be set to an undefined state (110 to 111), program behavior would become unpredictable.

Means for safeguarding this problem include:

- Mask off any unused bits of the variable. In the above example, ANDLW b’00000111’ will ensure that only the lower 3 bits of the number contain a value.
- Add extra cases to ensure that there will always be a known jump. For example in this case, two extra states must be added and used as error or Reset states.

CHAPTER 8

3V Tips ‘n Tricks

Table Of Contents

TIPS ‘N TRICKS INTRODUCTION

Overview - the 3.3 Volt to 5 Volt Connection

One of the by-products of our ever increasing need for processing speed is the steady reduction in the size of the transistors used to build microcontrollers. Up-integration at cheaper cost also drives the need for smaller geometries. With reduced size comes a reduction in the transistor breakdown voltage, and ultimately, a reduction in the supply voltage when the breakdown voltage falls below the supply voltage. So, as speeds increase and complexity mounts, it is an inevitable consequence that the supply voltages would drop from 5V to 3.3V, or even 1.8V for high density devices.

Microchip microcontrollers have reached a sufficient level of speed and complexity that they too are making the transition to sub-5V supply voltages. The challenge is that most of the interface circuitry is still designed for 5V supplies. This means that, as designers, we now face the task of interfacing 3.3V and 5V systems. Further, the task includes not only logic level translation, but also powering the 3.3V systems and translating analog signals across the 3.3V/5V barrier.

This Tips ‘n Tricks book addresses these challenges with a collection of power supply building blocks, digital level translation blocks and even analog translation blocks. Throughout the book, multiple options are presented for each of the transitions, spanning the range from all-in-one interface devices, to low-cost discrete solutions. In short, all the blocks a designer is likely to need for handling the 3.3V challenge, whether the driving force is complexity, cost or size.

Additional information can be found on the Microchip web site at www.microchip.com/3volts.

> **Note:** The tips ‘n tricks presented here assume a 3.3V supply. However, the techniques work equally well for other supply voltages with the appropriate modifications.

Power Supplies

One of the first 3.3V challenges is generating the 3.3V supply voltage. Given that we are discussing interfacing 5V systems to 3.3V systems, we can assume that we have a stable 5 V_{DC} supply. This section will present voltage regulator solutions designed for the 5V to 3.3V transition. A design with only modest current requirements may use a simple linear regulator. Higher current needs may dictate a switching regulator solution. Cost sensitive applications may need the simplicity of a discrete diode regulator. Examples from each of these areas are included here, with the necessary support information to adapt to a wide variety of end applications.

Table 1: Power Supply Comparisons

Method	V_{REG}	I_Q	Eff.	Size	Cost	Transient Response
Zener Shun Reg.	10% Typ	5 mA	60%	Sm	Low	Poor
Series Linear Reg.	0.4% Typ	1 µA to 100 µA	60%	Sm	Med	Excellent
Switching Buck Reg.	0.4% Typ	30 µA to 2 mA	93%	Med to Lrg	High	Good