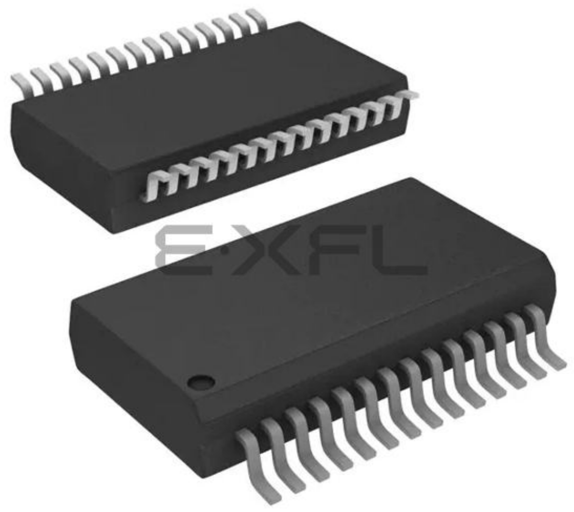


What is "Embedded - Microcontrollers"?

"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	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SSOP (0.209", 5.30mm Width)
Supplier Device Package	28-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16c63a-04-ss

Tip #13.2 Reading a Sensor With Higher Accuracy – Charge Balancing Method

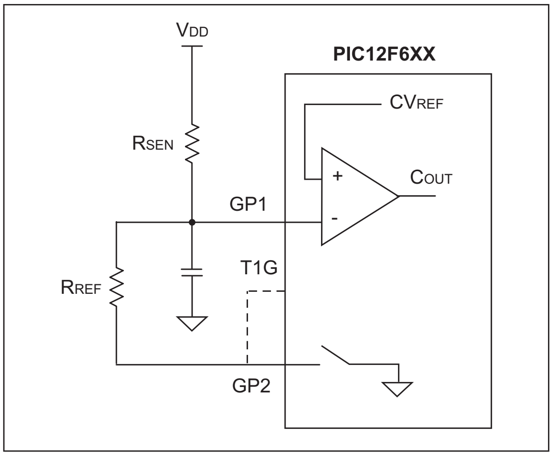
1. Sensor charges a capacitor
2. Reference resistor discharges the capacitor
3. Modulate reference resistor to maintain constant average charge in the capacitor
4. Use comparator to determine modulation

To improve resolution beyond 10 or 12 bits, a technique called "Charge Balancing" can be used. The basic concept is for the MCU to maintain a constant voltage on a capacitor by either allowing the charge to build through a sensor or discharge through a reference resistor. A timer is used to sample the capacitor voltage on regular intervals until a predetermined number of samples are counted. By counting the number of times the capacitor voltage is over an arbitrary threshold, the sensor voltage is determined.

The comparator and comparator voltage reference (CV_{REF}) on the PIC12F629/675 are ideal for this application.

1. GP1 average voltage = CV_{REF}
2. Time base as sampling rate
3. At the end of each time base period:
   - If GP1 > CV_{REF}, then GP2 Output Low
   - If GP1 < CV_{REF}, then GP2 Input mode
4. Accumulate the GP2 lows over many samples
5. Number of samples determines resolution
6. Number of GP2 lows determine effective duty cycle of R_{REF}

Figure 13-3



Tip #13.3 Reading a Sensor With Higher Accuracy – A/D Method

NTC (Negative Temperature Coefficient) sensors have a non-linear response to temperature changes. As the temperature drops, the amount the resistance changes becomes less and less. Such sensors have a limited useful range because the resolution becomes smaller than the A/D resolution as the temperature drops. By changing the voltage divider of the R_{SEN}, the temperature range can be expanded.

To select the higher temperature range, GP1 outputs ‘1’ and GP2 is set as an input. For the lower range, GP2 outputs ‘1’ and GP1 is configured as an input. The lower range will increase the amount the sensor voltage changes as the temperature drops to allow a larger usable sensor range.

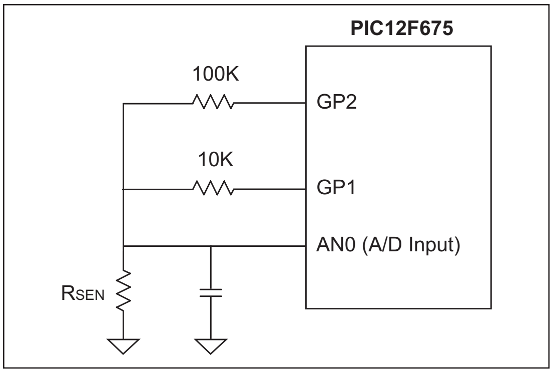
Summary:

High range: GP1 output ‘1’ and GP2 input

Low range: GP1 input and GP2 output ‘1’

1. 10K and 100K resistors are used to set the range
2. V_{REF} for A/D = V_{DD}
3. Rth calculation is independent of V_{DD}
4. Count = $R_{SEN}/(R_{SEN}+R_{REF})$ x 255
5. Don’t forget to allow acquisition time for the A/D

Figure 13-4

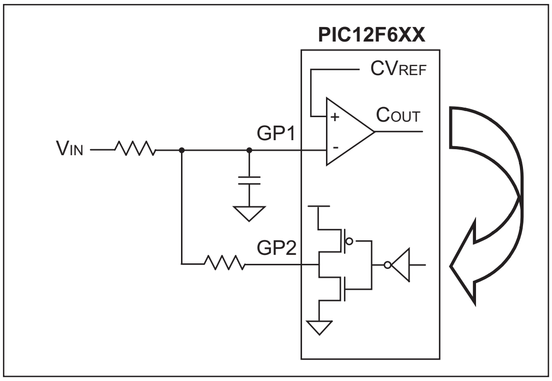


TIP #14 Delta-Sigma Converter

The charge on the capacitor on GP1 is maintained about equal to the CV_{REF} by the MCU monitoring C_{OUT} and switching GP2 from Input mode or output low appropriately. A timer is used to sample the C_{OUT} bit on a periodic basis. Each time GP2 is driven low, a counter is incremented. This counter value corresponds to the input voltage.

To minimize the affects of external component tolerances, temperature, etc., the circuit can be calibrated. Apply a known voltage to the input and allow the microcontroller to count samples until the expected result is calculated. By taking the same number of samples for subsequent measurements, they become calibrated measurements.

Figure 14-1



1. GP1 average voltage = CV_{REF}
2. Time base as sampling rate
3. At the end of each time base period:
   - If GP1 > CV_{REF}, then GP2 Output Low
   - If GP1 < CV_{REF}, then GP2 Output High
4. Accumulate the GP2 lows over many samples
5. Number of samples determines resolution

TIP #18 Swap File Register with W

Example 18-1

```
        SWAPWF        MACRO  REG
                      XORWF  REG,F
                      XORWF  REG,W
                      XORWF  REG,F
                      ENDM
```

The following macro swaps the contents of W and REG without using a second register.

Needs: 0 TEMP registers
3 Instructions
3 TCY

An efficient way of swapping the contents of a register with the working register is to use three XORWF instructions. It requires no temporary registers and three instructions. Here’s an example:

W	REG	Instruction
10101100	01011100	XORWF REG,F
10101100	11110000	XORWF REG,W
01011100	11110000	XORWF REG,F
01011100	10101100	Result

TIP #19 Bit Shifting Using Carry Bit

Rotate a byte through carry without using RAM variable for loop count:

- Easily adapted to serial interface transmit routines.
- Carry bit is cleared (except last cycle) and the cycle repeats until the zero bit sets indicating the end.

Example 19-1

```
       LIST P=PIC12f629
       INCLUDE P12f629.INC
       buffer      equ   0x20

bsf     STATUS,C     ;Set 'end of loop' flag
rlf     buffer,f     ;Place first bit into C
bcf     GPIO,Dout    ;precondition output
btfsc   STATUS,C     ;Check data 0 or 1 ?
bsf     GPIO,Dout
bcf     STATUS,C     ;Clear data in C
rlf     buffer,f     ;Place next bit into C
movf    buffer,f     ;Force Z bit
btfss   STATUS,Z     ;Exit?
goto    Send_Loop
```

TIP #9 Two-Speed Start-Up

Two-speed startup is a useful feature on some nanoWatt and all nanoWatt XLP devices which helps reduce power consumption by allowing the device to wake up and return to sleep faster. Using the internal oscillator, the user can execute code while waiting for the Oscillator Start-up (OST) timer to expire (LP, XT or HS modes). This feature (called “Two-Speed Start-up”) is enabled using the IESO configuration bit. A Two-Speed Start-up will clock the device from an internal RC oscillator until the OST has expired. Switching to a faster internal oscillator frequency during start-up is also possible using the OSCCON register. The example below shows several stages on how this can be achieved. The number of frequency changes is dependent upon the designer’s discretion. Assume a 20 MHz crystal (HS Mode) in the PIC16F example below.

Example:

```
TCY
(Instruction Time)   Instruction
                     ORG   0x05            ;Reset vector

125 µs @ 32 kHz      BSF   STATUS,RP0      ;bank1
125 µs @ 32 kHz      BSF   OSCCON,IRCF2    ;switch to 1 MHz

4 µs @ 1 MHz         BSF   OSCCON,IRCF1    ;switch to 4 MHz

1 µs @ 4 MHz         BSF   OSCCON,IRCF0    ;switch to 8 MHz

500 ns               application code
500 ns               application code
…                    ….
..                   …
(eventually OST expires, 20 MHz crystal clocks the device)

200 ns               application code
…                    ….
..                   …
```

TIP #10 Clock Switching

Some nanoWatt devices and all nanoWatt XLP devices have multiple internal and external clock sources, as well as logic to allow switching between the available clock sources as the main system clock. This allows for significant power savings by choosing different clocks for different portions of code. For example, an application can use the slower internal oscillator when executing non-critical code and then switch to a fast high-accuracy oscillator for time or frequency sensitive code. Clock switching allows much more flexible applications than being stuck with a single clock source. Clock switching sequences vary by device family, so refer to device data sheets or Family Reference Manuals for the specific clock switching sequences.

TIP #11 Use Internal RC Oscillators

If frequency precision better than ±5% is not required, it is best to utilize the internal RC oscillators inside all nanoWatt and nanoWatt XLP devices. The internal RC oscillators have better frequency stability than external RC oscillators, and consume less power than external crystal oscillators. Additionally, the internal clock can be configured for many frequency ranges using the internal PLL module to increase frequency and the postscaler to reduce it. All these options can be configured in firmware.

TIP #14 Use `NOP` and Idle Mode

When waiting on a blocking loop (e.g. waiting for an interrupt), instead put the device into Idle mode to disable the CPU. The peripheral interrupt will wake up the device. Idle mode consumes much less current than constantly reading RAM and jumping back. If the CPU cannot be disabled because the loop required some calculations, such as incrementing a counter, instead of doing a very tight loop that loops many times, add `NOP`s into the loop. See the code example below. A `NOP` requires less current to execute than reading RAM or branching operations, so current can be reduced. The overall loop count can be adjusted to account for the extra instructions for the `NOP`s.

Example:

Replace:

```
while(!_T1IF);
```

with Idle mode:

```
IEC0bits.T1IE = 1;
Idle();
```

and replace:

```
while(!_T1IF){
    i++;
}
```

with extra NOP instructions:

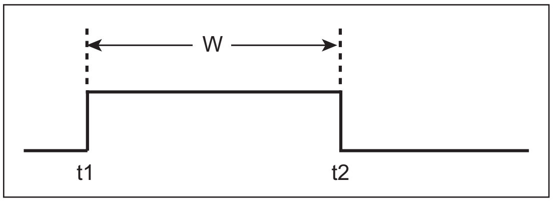
```
while(!_T1IF){
   i++;
    Nop();
    Nop();
    Nop();
    Nop();
    Nop();
}
```

TIP #15 Peripheral Module Disable (PMD) Bits

PIC24, dsPIC DSCs, and PIC32 devices have PMD bits that can be used to disable peripherals that will not be used in the application. Setting these bits disconnects all power to the module as well as SFRs for the module. Because power is completely removed, the PMD bits offer additional power savings over disabling the module by turning off the module's enable bit. These bits can be dynamically changed so that modules which are only used periodically can be disabled for the remainder of the application. The PMD bits are most effective at high clock speeds and when operating at full speed allowing the average power consumption to be significantly reduced.

TIP #3 Measuring Pulse Width

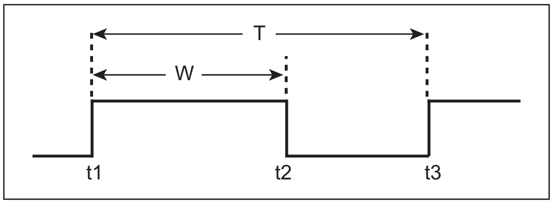
Figure 3-1: Pulse Width



1. Configure control bits CCPxM3:CCPxM0 (CCPxCON<3:0>) to capture every rising edge of the waveform.
2. Configure Timer1 prescaler so that Timer1 will run WMAX without overflowing.
3. Enable the CCP interrupt (CCPxIE bit).
4. When CCP interrupt occurs, save the captured timer value (t1) and reconfigure control bits to capture every falling edge.
5. When CCP interrupt occurs again, subtract saved value (t1) from current captured value (t2) – this result is the pulse width (W).
6. Reconfigure control bits to capture the next rising edge and start process all over again (repeat steps 3 through 6).

TIP #4 Measuring Duty Cycle

Figure 4-1: Duty Cycle



The duty cycle of a waveform is the ratio between the width of a pulse (W) and the period (T). Acceleration sensors, for example, vary the duty cycle of their outputs based on the acceleration acting on a system. The CCP module, configured in Capture mode, can be used to measure the duty cycle of these types of sensors. Here's how:

1. Configure control bits CCPxM3:CCPxM0 (CCPxCON<3:0>) to capture every rising edge of the waveform.
2. Configure Timer1 prescaler so that Timer1 will run TMAX(1) without overflowing.
3. Enable the CCP interrupt (CCPxIE bit).
4. When CCP interrupt occurs, save the captured timer value (t1) and reconfigure control bits to capture every falling edge.

> **Note 1:** TMAX is the maximum pulse period that will occur.

5. When the CCP interrupt occurs again, subtract saved value (t1) from current captured value (t2) – this result is the pulse width (W).
6. Reconfigure control bits to capture the next rising edge.
7. When the CCP interrupt occurs, subtract saved value (t1) from the current captured value (t3) – this is the period (T) of the waveform.
8. Divide T by W – this result is the Duty Cycle.
9. Repeat steps 4 through 8.

TIP #5 Measuring RPM Using an Encoder

Revolutions Per Minute (RPM), or how fast something turns, can be sensed in a variety of ways. Two of the most common sensors used to determine RPM are optical encoders and Hall effect sensors. Optical encoders detect the presence of light shining through a slotted wheel mounted to a turning shaft (see Figure 5-1.) As the shaft turns, the slots in the wheel pass by the eye of the optical encoder. Typically, an infrared source on the other side of the wheel emits light that is seen by the optical encoder through slots in the wheel. Hall effect sensors work by sensing the position of the magnets in an electric motor, or by sensing a permanent magnet mounted to a rotating object (see Figure 5-2). These sensors output one or more pulses per revolution (depending on the sensor).

Figure 5-1: Optical Encoder

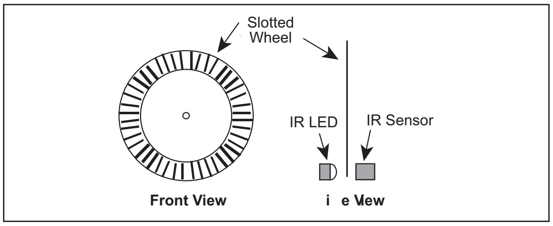
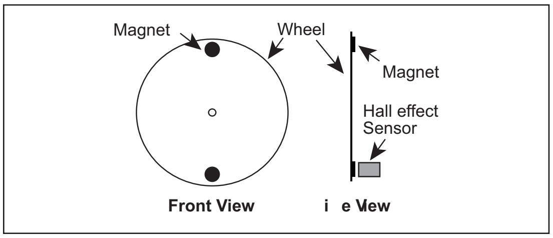


Figure 5-2: Hall Effect Sensor



In Figure 5-3 and Figure 5-4, the waveform is high when light is passing through a slot in the encoder wheel and shining on the optical sensor. In the case of a Hall effect sensor, the high corresponds to the time that the magnet is in front of the sensor. These figures show the difference in the waveforms for varying RPMs. Notice that as RPM increases, the period (T) and pulse width (W) becomes smaller. Both period and pulse width are proportional to RPM. However, since the period is the greater of the two intervals, it is good practice to measure the period so that the RPM reading from the sensor will have the best resolution. See Tip #1 for measuring period. The technique for measuring period with averaging described in Tip #2 is useful for measuring high RPMs.

Figure 5-3: Low RPM

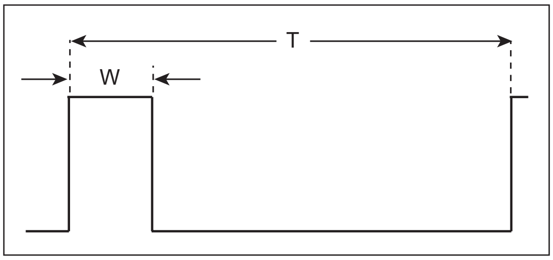
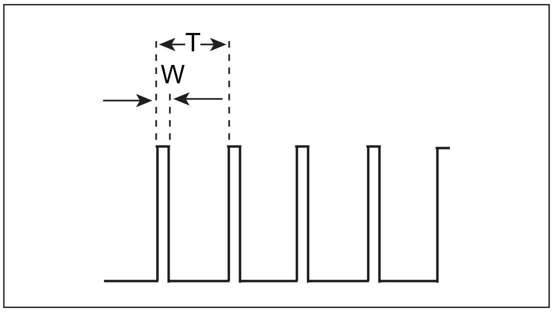


Figure 5-4: High RPM



TIP #8 Modulation Formats

The CCP module, configured in Compare mode, can be used to generate a variety of modulation formats. The following figures show four commonly used modulation formats:

Figure 8-1: Pulse-width Modulation

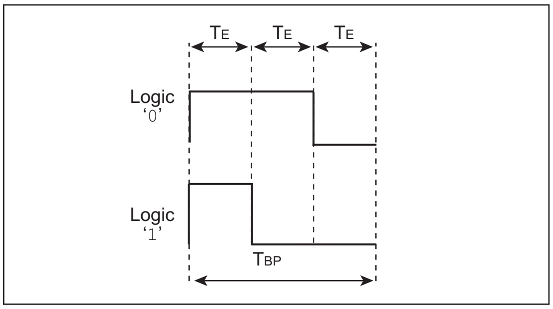


Figure 8-2: Manchester

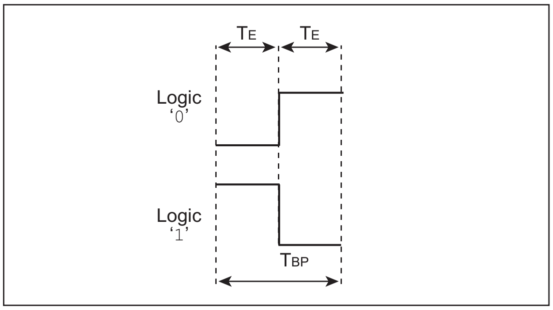


Figure 8-3: Pulse Position Modulation

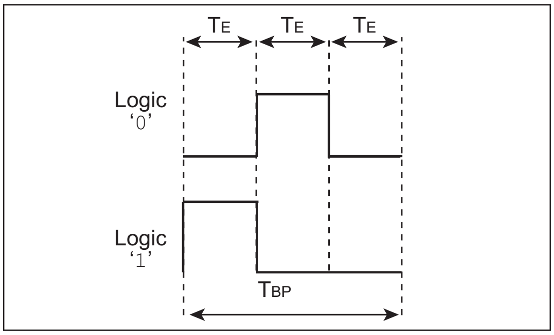
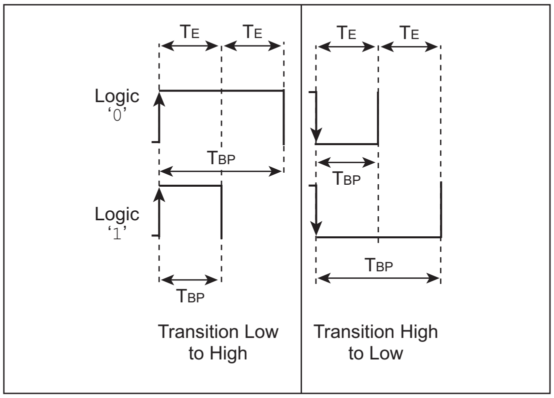


Figure 8-4: Variable Pulse-width Modulation



The figures show what a logic '0' or a logic '1' looks like for each modulation format. A transmission typically resembles an asynchronous serial transmission consisting of a Start bit, followed by 8 data bits, and a Stop bit.

T_E is the basic timing element in each modulation format and will vary based on the desired baud rate.

Trigger Special Event mode can be used to generate T_E, (the basic timing element). When the CCPx interrupt is generated, code in the ISR routine would implement the desired modulation format (additional modulation formats are also possible).

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

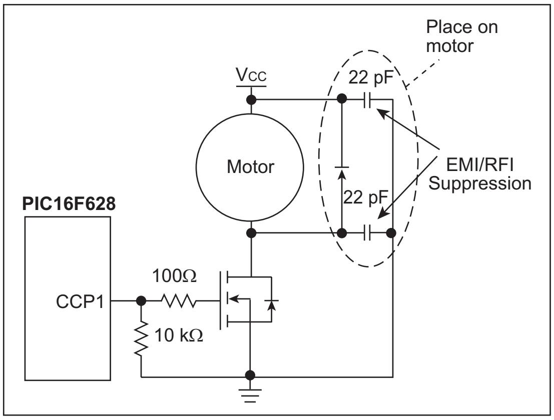


Figure 14-1 shows a unidirectional speed controller circuit for a brushed DC motor. Motor speed is proportional to the duty cycle of the PWM output on the CCP1 pin. The following steps show how to configure the PIC16F628 to generate a 20 kHz PWM with 50% duty cycle. The microcontroller is running on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

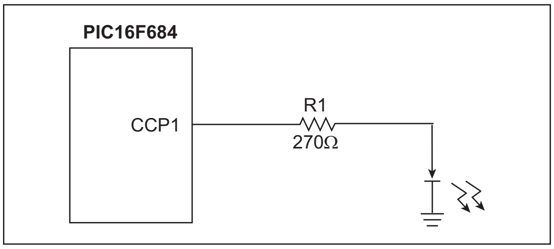
Step #4: Configure CCP1CON

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

TIP #18 Varying LED Intensity

The intensity of an LED can be varied by pulse-width modulating the voltage across the LED. A microcontroller typically drives an LED with the circuit shown in Figure 18-1. The purpose of R1 is to limit the LED current so that the LED runs in its specified current and voltage range, typically around 1.4 volts at 20 mA. Modulating the LED drive pin on the microcontroller will vary the average current seen by the LED and thus its intensity. As mentioned in Tip #13, LEDs and other light sources should be modulated at no less than 100 Hz in order to prevent noticeable flicker.

Figure 18-1: LED Drive

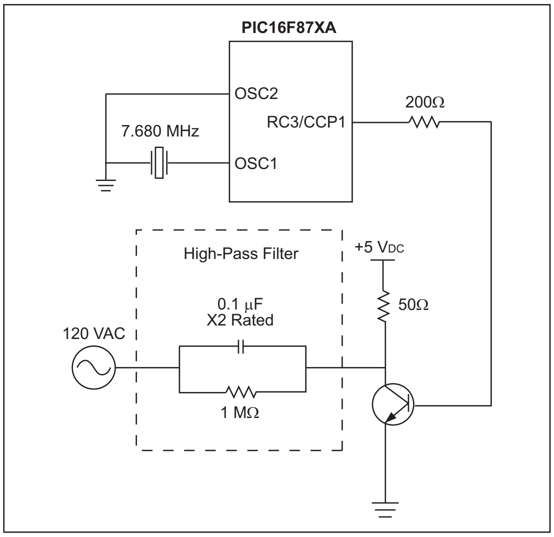


The CCP module, configured in PWM mode, is ideal for varying the intensity of an LED. Adjustments to the intensity of the LED are made by simply varying the duty cycle of the PWM signal driving the LED. This is accomplished by varying the CCPRxL register between 0 and 0xFF.

TIP #19 Generating X-10® Carrier Frequency

X-10 uses a piggybacked 120 kHz square wave (at 50% duty cycle) to transmit information over 60 Hz power lines. The CCP module, running in PWM mode, can accurately create the 120 kHz square wave, referred to as the carrier frequency. Figure 19-1 shows how the 120 kHz carrier frequency is piggybacked onto the sinusoidal 60 Hz power waveform.

Figure 19-1: Carrier Frequency With Sinusoidal Waveform



X-10 specifies the carrier frequency at 120 kHz (± 2 kHz). The system oscillator in Figure 18-1 is chosen to be 7.680 MHz, so that the CCP module can generate precisely 120 kHz. X-10 requires that the carrier frequency be turned on and off at different points on the 60 Hz power waveform. This is accomplished by configuring the TRIS register for the CCP1 pin as either an input (carrier frequency off) or an output (carrier frequency on). Refer to Application Note AN236 "*X-10 Home Automation Using the PIC16F877A*" for more details on X-10 and for source code for setting up the CCP module appropriately.

TIP #7 One-Shot

When dealing with short duration signals or glitches, it is often convenient to stretch out the event using a mono-stable, multi-vibrator or one-shot. Whenever the input pulses, the one-shot fires holding its output for a preset period of time. This stretches the short trigger input into a long output which the microcontroller can capture.

The circuit is designed with two feedback paths around a comparator. The first is a positive hysteresis feedback which sets a two level threshold, V_{HI} and V_{LO}, based on the state of the comparator output. The second feedback path is an RC time circuit.

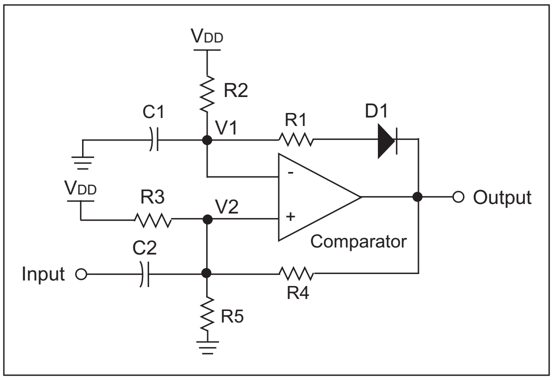
The one-shot circuit presented in Figure 7-1 is triggered by a low-high transition on its input and generates a high output pulse. Using the component values from the example, the circuit's operation is as follows.

Prior to triggering, C1 will have charged to a voltage slightly above 0.7V due to resistor R2 and D1 (R1 << R2 and will have only a minimal effect on the voltage). The comparator output will be low, holding the non-inverting input slightly below 0.7V due to the hysteresis feedback through R3, R4 and R5 (the hysteresis lower limit is designed to be less than 0.7V). With the non-inverting input held low, C2 will charge up to the difference between the circuit input and the voltage present at the non-inverting input.

When the circuit input is pulsed high, the voltage present at the non-inverting input is pulled above 0.7V due to the charge in C2. This causes the output of the comparator to go high, the hysteresis voltage at the non-inverting input goes to the high threshold voltage, and C1 begins charging through R2.

When the voltage across C1 exceeds the high threshold voltage, the output of the comparator goes low, C1 is discharged to just above the 0.7V limit, the non-inverting input is pulled below 0.7V, and the circuit is reset for the next pulse input, waiting for the next trigger input.

Figure 7-1: One-Shot Circuit



To design the one-shot, first create the hysteresis feedback using the techniques from Tip #3. Remember to set the low threshold below 0.7V. Next, choose values for R2 and C1 using Equation 7-1.

Equation 7-1

$$T_{PULSE} = \frac{R2 * C1 * \ln(V_{TH}/V_{TL})}{4}$$

D1 can be any low voltage switching diode. R1 should be 1% to 2% of R2 and C2 should be between 100 and 220 pF.

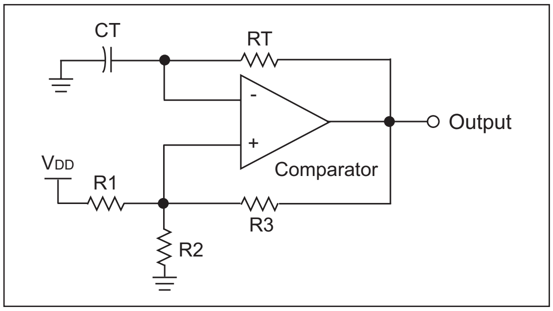
Example:

- V_{DD} = 5V, V_{TH} = 3.0V, V_{TL} = 2.5V
- From Tip #3, R4 = 1k, R5 = 1.5k and R3 = 12k
- T_{PULSE} = IMS, C1 = .1 μF and R2 = 15k
- D1 is a 1N4148, R1 = 220 and C2 = 150 pF

TIP #8 Multi-Vibrator (Square Wave Output)

A multi-vibrator is an oscillator designed around a voltage comparator or operational amplifier (see Figure 8-1). Resistors R1 through R3 form a hysteresis feedback path from the output to the non-inverting input. Resistor RT and capacitor CT form a time delay network between the output and the inverting input. At the start of the cycle, CT is discharged holding the non-inverting input at ground, forcing the output high. A high output forces the non-inverting input to the high threshold voltage (see Tip #3) and charges CT through RT. When the voltage across CT reaches the high threshold voltage, the output is forced low. A low output drops the non-inverting input to the low threshold voltage and discharges CT through RT. When the voltage across CT reaches the low threshold voltage, the output is forced high and the cycle starts over.

Figure 8-1: Multi-Vibrator Circuit



To design a multi-vibrator, first design the hysteresis feedback path using the procedure in Tip #3. Be careful to choose threshold voltages (V_{TH} and V_{TL}) that are evenly spaced within the common mode range of the comparator and centered on $V_{DD}/2$. Then use V_{DD} and V_{TL} to calculate values for RT and CT that will result in the desired oscillation frequency F_{OSC}. Equation 8-1 defines the relationship between RT, CT, V_{TH}, V_{TL} and F_{OSC}.

Equation 8-1

$$F_{OSC} = \frac{1}{2 * RT * CT * \ln(V_{TH}/V_{TL})}$$

Example:

- $V_{DD} = 5V$, $V_{TH} = 3.333$, $V_{TL} = 1.666V$
- R1, to R2, to R3 = 10k
- $R_T = 15$ kHz, $C_T = .1\ \mu F$ for $F_{OSC} = 480$ Hz

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

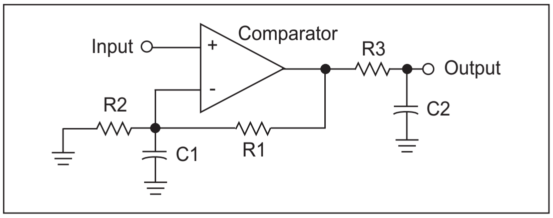
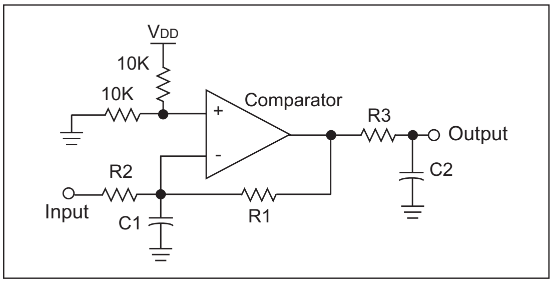


Figure 12-2: Inverting Amplifier



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

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

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

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

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

$$\text{Gain} = \frac{R_1}{R_2}$$

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:

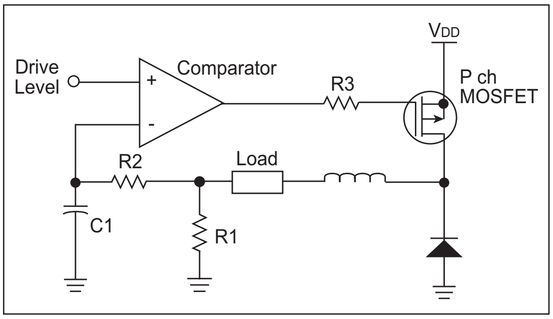
- For C2 will set the corner F
- Gain = 6.156, R1 = R3 = 19.8k
- R2 = 3.84k, C1 = .047 μF, F_{CORNER} = 171 Hz
- C2 = .22 μF

TIP #13 PWM High-Current Driver

This tip combines a comparator with a MOSFET transistor and an inductor to create a switch mode high-current driver circuit. (See Figure 13-1).

The operation of the circuit begins with the MOSFET off and no current flowing in the inductor and load. With the sense voltage across R1 equal to zero and a DC voltage present at the drive level input, the output of the comparator goes low. The low output turns on the MOSFET and a ramping current builds through the MOSFET, inductor, load and R1.

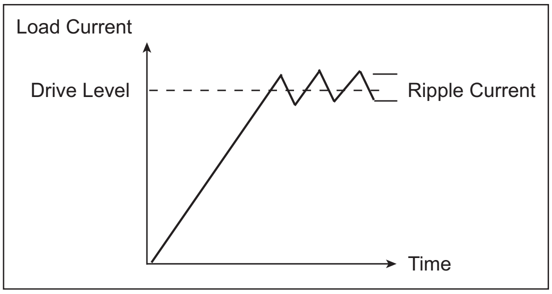
Figure 13-1: High Current Driver



When the current ramps high enough to generate a voltage across R1 equal to the drive level, the comparator output goes high turning off the MOSFET. The voltage at the junction of the MOSFET and the inductor then drops until D1 forward biases. The current continues ramping down from its peak level toward zero. When the voltage across the sense resistor R1 drops below the drive level, the comparator output goes low, the MOSFET turns on, and the cycle starts over.

R2 and C1 form a time delay network that limits the switching speed of the driver and causes it to slightly overshoot and undershoot the drive level when operating. The limit is necessary to keep the switching speed low, so the MOSFET switches efficiently. If R2 and C1 were not present, the system would run at a speed set by the comparator propagation delay and the switching speed of the MOSFET. At that speed, the switching time of the MOSFET would be a significant portion of the switching time and the switching efficiency of the MOSFET would be too low.

Figure 13-1: Current Through the Load



To design a PWM high current driver, first determine a switching speed (F_{SWX}) that is appropriate for the system. Next, choose a MOSFET and D1 capable of handling the load current requirements. Then choose values for R2 and C1 using Equation 13-1.

Equation 13-1

$$F_{SWX} = \frac{2}{R2 * C1}$$

Next determine the maximum ripple current that the load will tolerate, and calculate the required inductance value for L1 using Equation 13-2.

Equation 13-2

$$L = \frac{V_{DD} - V_{LOAD}}{I_{RIPPLE} * F_{SWX} * 2}$$

Finally, choose a value for R1 that will produce a feedback ripple voltage of 100 mV for the maximum ripple current I_{RIPPLE}.

Example:

- F_{SWX} = 10 kHz, R2 = 22k, C1 = .01 μF
- I_{RIPPLE} = 100 mA, V_{DD} = 12V, V_L = 3.5V
- L = 4.25 mH

TIP #17 Logic: AND/NAND Gate

This tip shows the use of the comparator to implement an AND gate and its complement the NAND gate (see Figure 17-2). Resistors R1 and R2 drive the non-inverting input with 2/3 the supply voltage. Resistors R3 and R4 average the voltage of input A and B at the inverting input. If either A or B is low, the average voltage will be one half V_{DD} and the output of the comparator remains low. The output will go high only if both inputs A and B are high, which raises the input to the inverting input above 2/3 V_{DD}.

The operation of the NAND gate is identical to the AND gate, except that the output is inverted due to the swap of the inverting and non-inverting inputs.

Note: Typical propagation delay for the circuit is 250-350 ns using the typical on-chip comparator peripheral of a microcontroller. Delay measurements were made with 10k resistance values.

While the circuit is fairly simple, there are a few requirements for correct operation:

1. The inputs A and B must drive from ground to V_{DD} for the circuit to operate properly.
2. The combination of R1 and R2 will draw current constantly, so they must be kept large to minimize current draw.
3. All resistances on the inverting input react with the input capacitance of the comparator. So the speed of the gate will be affected by the source resistance of A and B, as well as, the size of resistors R3 and R4.
4. Resistor R2 must be 2 x R1.
5. Resistor R3 must be equal to R4.

Figure 17-1: AND Gate

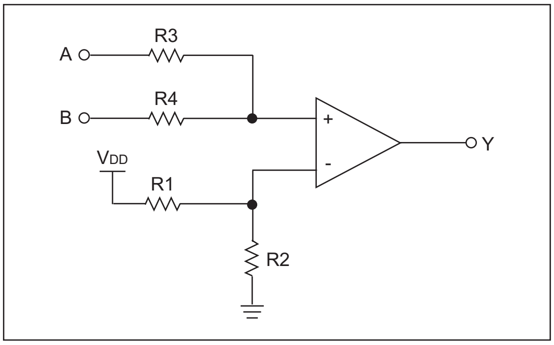
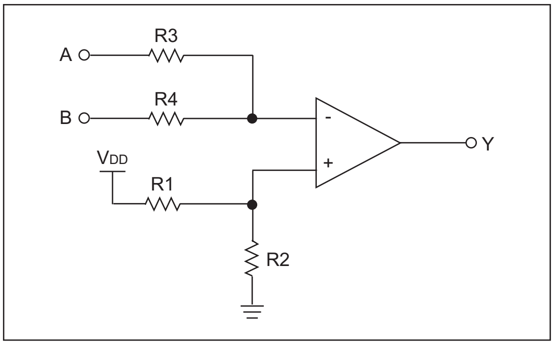


Figure 17-2: NAND Gate



Example:

- V_{DD} = 5V, R3 = R4 = 10k
- R1 = 5.1k, R2 = 10k

CHAPTER 5
DC Motor Control Tips ‘n Tricks

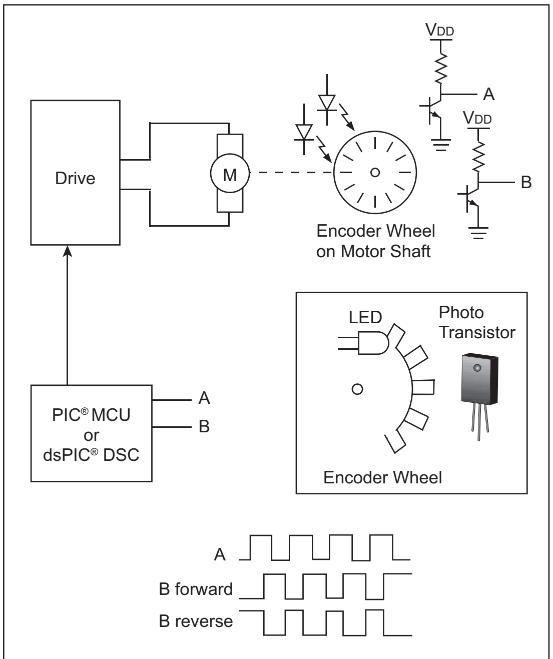
Table Of Contents

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).

Figure 7-2: Optical Speed/Direction/Position Sensing



Note: Frequency of one signal provides RPM of motor. Pulse count provides motor position. A-B phase provides motor direction.

Quadrature sensing can easily be accomplished in software, but there is generally an upper limit to the RPM. By using a few gates, the sensing can be done partially in hardware and partially in software. The new PIC18FXX31 and dsPIC 16-bit Digital Signal Controller families include an encoder interface that allows MUCH higher RPM motors to be measured with an excellent degree of accuracy.

Older Methods of Motor Sensing

Resolvers and analog tachometers are two older technologies for motor position/velocity sensing. An analog tachometer is simply an electric generator with a linear output over a specified range of RPM's. By knowing the output characteristics, the RPM can be known by simply measuring the voltage across the tachometer terminals.

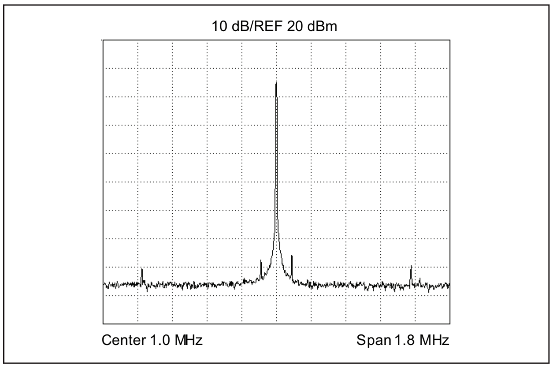
A resolver is a pair of coils that are excited by an external AC signal. The two coils are at 90° to each other so they pick up the AC signal at different strengths, depending on their orientation. The result is a sine or cosine output related to the angle of the resolver in reference to the AC signal. Inverse cosine/sine will produce the angle of the sensor. This type of sensor can be very accurate and is still used where absolute position must be known.

TIP #4 Creating a Dithered PWM Clock

In order to meet emissions requirements as mandated by the FCC and other regulatory organizations, the switching frequency of a power supply can be varied. Switching at a fixed frequency produces energy at that frequency. By varying the switching frequency, the energy is spread out over a wider range and the resulting magnitude of the emitted energy at each individual frequency is lower.

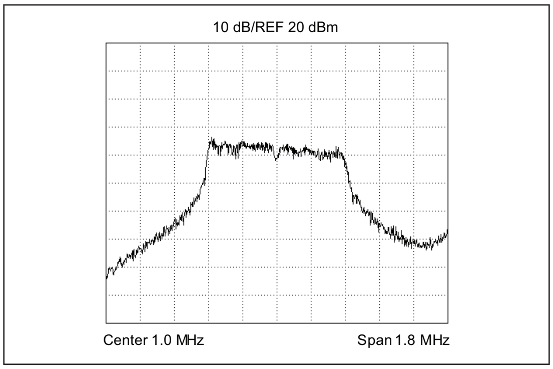
The PIC10F200 has an internal 4 MHz oscillator. A scaled version of oscillator can be output on a pin (Fosc/4). The scaled output is 1/4 of the oscillator frequency (1 MHz) and will always have a 50% duty cycle. Figure 4-1 shows a spectrum analyzer shot of the output of the Fosc/4 output.

Figure 4-1: Spectrum of Clock Output Before Dithering



The PIC10F200 provides an Oscillator Calibration (OSCCAL) register that is used to calibrate the frequency of the oscillator. By varying the value of the OSCCAL setting, the frequency of the clock output can be varied. A pseudo-random sequence was used to vary the OSCCAL setting, allowing frequencies from approximately 600 kHz to 1.2 MHz. The resulting spectrum is shown in Figure 4-2.

Figure 4-2: Spectrum of Clock Output After Dithering



By spreading the energy over a wider range of frequencies, a drop of more than 20 dB is achieved.

Example software is provided for the PIC10F200 that performs the pseudo-random sequence generation and loads the OSCCAL register.

The A/D converter is used to sense the output voltage for this particular application, V_{DD} is used as the reference to the A/D converter. If desired, a more accurate reference could be used. The output voltage is subtracted from the desired value, creating an error value.

This error becomes the input to the PID routine. The PID routine uses the error voltage to determine the appropriate duty cycle for the output drive. The PID constants are weighted so that the main portion of the control is proportional and integral. The differential component is not essential to this system and is not used. Furthermore, the PID constants could be optimized if a particular type of transient response was desired, or if a predictable transient load was to be connected.

Finally, the CCP module is used to create a PWM signal at the chosen frequency with the proper duty cycle.

Example software is provided for the PIC12F683 using the schematic in Figure 16-1.

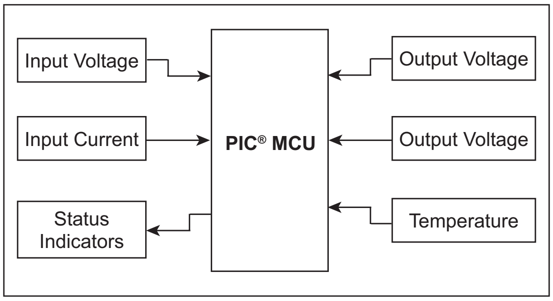
The following application notes are related to PID control algorithms and all include example software:

- AN258, “*Low Cost USB Microcontroller Programmer The Building of the PICkit® 1 Flash Starter Kit*” (DS00258)
- AN937, “*Implementing a PID Controller Using a PIC18 MCU*” (DS00937)
- AN964, “*Software PID Control of an Inverted Pendulum Using the PIC16F684*” (DS00937)

TIP #17 An Error Detection and Restart Controller

An error detection and restart controller can be created by combining Tip #18 and Tip #19. The controller uses the PIC microcontroller (MCU) Analog-to-Digital Converter (ADC) for making voltage and current measurements. Input voltage, input current, output voltage, output current, temperature and more can all be measured using the A/D converter. The on-board comparators are used for monitoring faster signals, such as output current, ensuring that they do not exceed maximum allowable levels. Many PIC MCUs have internal programmable comparator references, simplifying the circuit.

Figure 17-1: Block Diagram



Using a PIC MCU as a controller allows for a greater level of intelligence in system monitoring. Rather than a single event causing a shutdown, a combination of events can cause a shutdown. A certain number of events in a certain time frame or possibly a certain sequence of events could be responsible for a shutdown.

The PIC MCU has the ability to restart the supply based on the shutdown event. Some events (such as overcurrent) may call for immediate restart, while other events (such as overtemperature) may require a delay before restarting, perhaps monitoring other parameters and using those to determine when to restart.

It is also possible to build this type of error detection and restart controller into many of the tips listed within this guide.

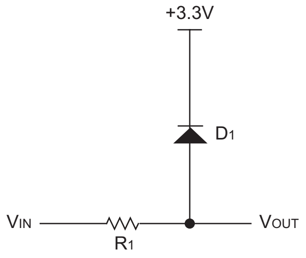
TIP #17 5V → 3V Analog Limiter

When moving a 5V signal down to a 3.3V system, it is sometimes possible to use the attenuation as gain. If the desired signal is less than 5V, then attaching that signal to a 3.3V ADC will result in larger conversion values. The danger is when the signal runs to the 5V rail. A method is therefore required to control the out-of-range voltages while leaving the in-range voltages unaffected. Three ways to accomplish this will be discussed here.

1. Using a diode to clamp the overvoltage to the 3.3V supply.
2. Using a Zener diode to clamp the voltage to any desired limit.
3. Using an op amp with a diode to perform a precision clamp.

The simplest method to perform the overvoltage clamp is identical to the simple method of interfacing a 5V digital signal to the 3.3V digital signals. A resistor and a diode are used to direct excess current into the 3.3V supply. The resistor must be sized to protect the diode and the 3.3V supply while not adversely affecting the analog performance. If the impedance of the 3.3V supply is too low, then this type of clamp can cause the 3.3V supply voltage to increase. Even if the 3.3V supply has a good low-impedance, this type of clamp will allow the input signal to add noise to the 3.3V supply when the diode is conducting and if the frequency is high enough, even when the diode is not conducting due to the parasitic capacitance across the diode.

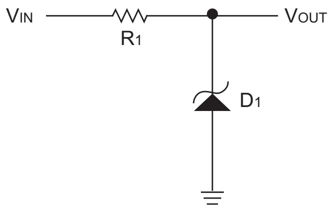
Figure 17-1: Diode Clamp



$V_{OUT} = 3.3V + V_F$ if $V_{IN} > 3.3V + V_F$
$V_{OUT} = V_{IN}$ if $V_{IN} \leq 3.3V + V_F$
V_F is the forward drop of the diode.

To prevent the input signal from affecting the supply or to make the input more robust to larger transients, a variation is to use a Zener diode. The Zener diode is slower than the fast signal diode typically used in the first circuit. However, they are generally more robust and do not rely on the characteristics of the power supply to perform the clamping. The amount of clamping they provide is dependant upon the current through the diode. This is set by the value of R1. R1 may not be required if the output impedance of the VIN source is sufficiently large.

Figure 17-2: Zener Clamp



$V_{OUT} = V_{BR}$ if $V_{IN} > V_{BR}$
$V_{OUT} = V_{IN}$ if $V_{IN} \leq V_{BR}$
V_{BR} is the reverse breakdown voltage of the Zener diode.