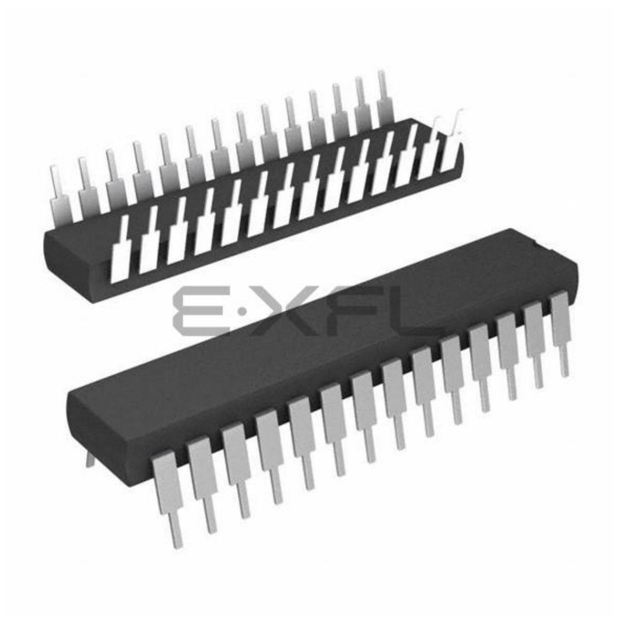


Welcome to E-XFL.COM

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	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	20MHz
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	Through Hole
Package / Case	28-DIP (0.300", 7.62mm)
Supplier Device Package	28-SPDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16c63a-20e-sp

CHAPTER 1
8-Pin Flash PIC® Microcontrollers 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 8-pin Flash PIC® microcontrollers (MCU) are used in an wide range of everyday products, from toothbrushes, hair dryers and rice cookers to industrial, automotive and medical products.

The PIC12F629/675 MCUs merge all the advantages of the PIC MCU architecture and the flexibility of Flash program memory into an 8-pin package. They provide the features and intelligence not previously available due to cost and board space limitations. Features include a 14-bit instruction set, small footprint package, a wide operating voltage of 2.0 to 5.5 volts, an internal programmable 4 MHz oscillator, on-board EEPROM data memory, on-chip voltage reference and up to 4 channels of 10-bit A/D. The flexibility of Flash and an excellent development tool suite, including a low-cost In-Circuit Debugger, In-Circuit Serial Programming™ and MPLAB® ICE 2000 emulation, make these devices ideal for just about any embedded control application.

TIPS 'N TRICKS WITH HARDWARE

The following series of Tips 'n Tricks can be applied to a variety of applications to help make the most of the 8-pin dynamics.

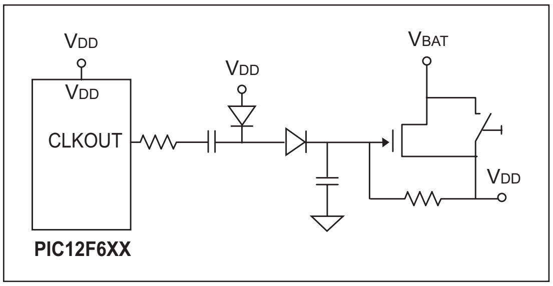
TIP #11 V_{DD} Self Starting Circuit

Building on the previous topic, the same charge pump can be used by the MCU to supply its own V_{DD}. Before the switch is pressed, V_{BAT} has power and the V_{DD} points are connected together but unpowered. When the button is pressed, power is supplied to V_{DD} and the MCUs CLKOUT (in external RC oscillator mode) begins toggle. The voltage generated by the charge pump turns on the FET allowing V_{DD} to remain powered. To power down the MCU, execute a Sleep instruction. This allows the MCU to switch off its power source via software.

Advantages:

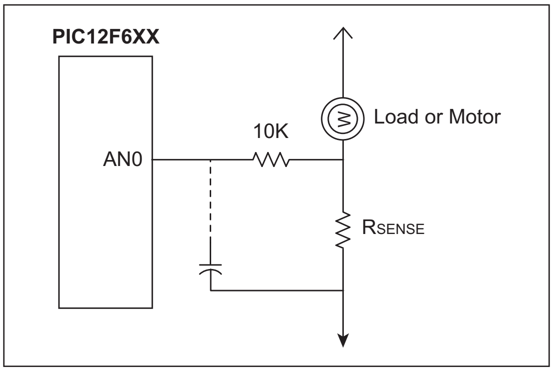
- PIC MCU leakage current nearly 0
- Low cost (uses n-channel FET)
- Reliable
- No additional I/O pins required

Figure 11-1



TIP #12 Using PIC® MCU A/D For Smart Current Limiter

Figure 12-1



- Detect current through low side sense resistor
- Optional peak filter capacitor
- Varying levels of overcurrent response can be realized in software

By adding a resistor (R_{SENSE}) in series with a motor, the A/D can be used to measure in-rush current, provide current limiting, over-current recovery or work as a smart circuit breaker. The 10K resistor limits the analog channel current and does not violate the source impedance limit of the A/D.

TIP #16 Optimizing Destinations

- Destination bit determines W for F for result
- Look at data movement and restructure

Example 16-1

Example: A + B → A

```
MOVF      A,W
ADDWF     B,W
MOVWF     A
```

3 instructions

```
MOVF    B,W
ADDWF   A,F
```

2 instructions

Careful use of the destination bits in instructions can save program memory. Here, register A and register B are summed and the result is put into the A register. A destination option is available for logic and arithmetic operations. In the first example, the result of the ADDWF instruction is placed in the working register. A MOVWF instruction is used to move the result from the working register to register A. In the second example, the ADDWF instruction uses the destination bit to place the result into the A register, saving an instruction.

TIP #17 Conditional Bit Set/Clear

- To move single bit of data from REGA to REGB
- Precondition REGB bit
- Test REGA bit and fix REGB if necessary

Example 17-1

```
BTFSS  REGA,2
BCF    REGB,5
BTFSC  REGA,2
BSF    REGB,5
```

4 instructions

```
BCF    REGB,5
BTFSC  REGA,2
BSF    REGB,5
```

3 instructions

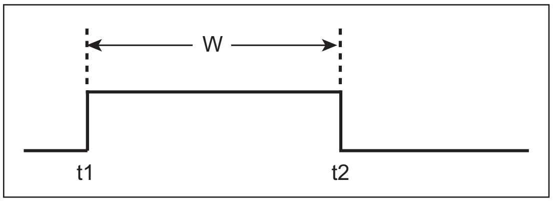
One technique for moving one bit from the REGA register to REGB is to perform bit tests. In the first example, the bit in REGA is tested using a BTFSS instruction. If the bit is clear, the BCF instruction is executed and clears the REGB bit, and if the bit is set, the instruction is skipped.The second bit test determines if the bit is set, and if so, will execute the BSF and set the REGB bit, otherwise the instruction is skipped. This sequence requires four instructions.

A more efficient technique is to assume the bit in REGA is clear, and clear the REGB bit, and test if the REGA bit is clear. If so, the assumption was correct and the BSF instruction is skipped, otherwise the REGB bit is set. The sequence in the second example uses three instructions because one bit test was not needed.

One important point is that the second example will create a two-cycle glitch if REGB is a port outputting a high. This is caused by the BCF and BTFSC instructions that will be executed regardless of the bit value in REGA.

TIP #3 Measuring Pulse Width

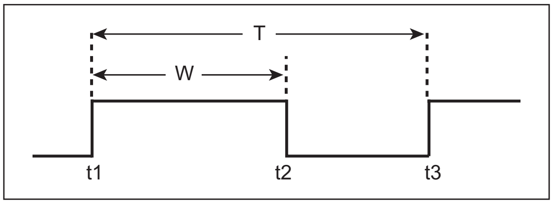
Figure 3-1: Pulse Width



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 W_{MAX} 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 T_{MAX}(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: T_{MAX} 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 #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

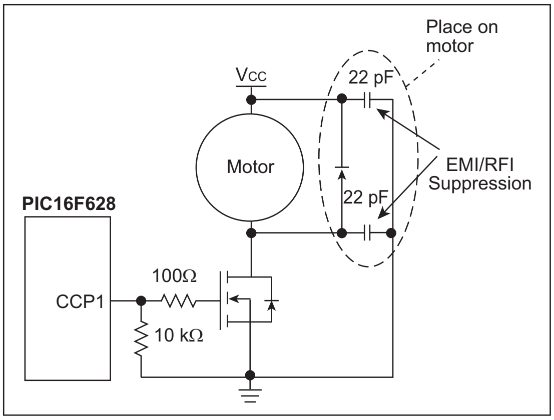


Figure 14-1 shows a unidirectional speed controller circuit for a brushed DC motor. Motor speed is proportional to the duty cycle of the PWM output on the CCP1 pin. The following steps show how to configure the PIC16F628 to generate a 20 kHz PWM with 50% duty cycle. The microcontroller is running on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

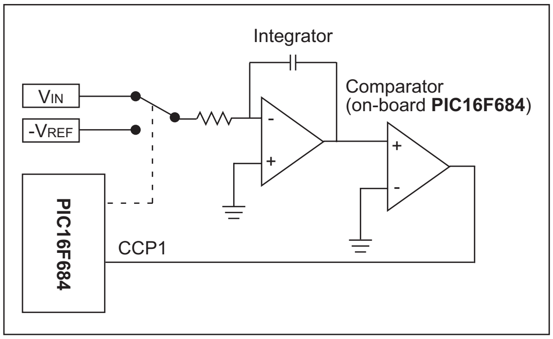
Step #4: Configure CCP1CON

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

TIP #21 Dual-Slope Analog-to-Digital Converter

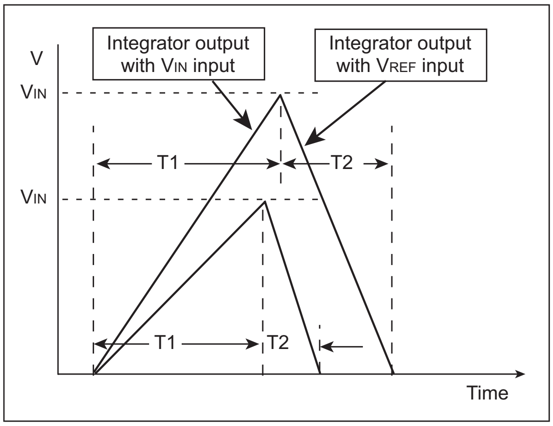
A circuit for performing dual-slope A/D conversion utilizing the CCP module is shown in Figure 21-1.

Figure 21-1: Dual-Slope Analog-to-Digital Converter



Dual-slope A/D conversion works by integrating the input signal (V_{IN}) for a fixed time (T1). The input is then switched to a negative reference ($-V_{REF}$) and integrated until the integrator output is zero (T2). V_{IN} is a function of V_{REF} and the ratio of T2 to T1.

Figure 21-2: V vs. Time



The components of this conversion type are the fixed time and the timing of the falling edge. The CCP module can accomplish both of these components via Compare mode and Capture mode respectively. Here’s how:

1. Configure the CCP module in Compare mode, Special Event Trigger.
2. Switch the analog input into the integrator from V_{REF} to V_{IN}.
3. Use the CCP module to wait T1 (T1 chosen based on capacitor value).
4. When the CCP interrupt occurs, switch the analog input into the regulator from V_{IN} to V_{REF} and reconfigure the module in Capture mode; wait for falling edge.
5. When the next CCP interrupt occurs, the time captured by the module is T2.
6. Calculate V_{IN} using Equation 21-1.

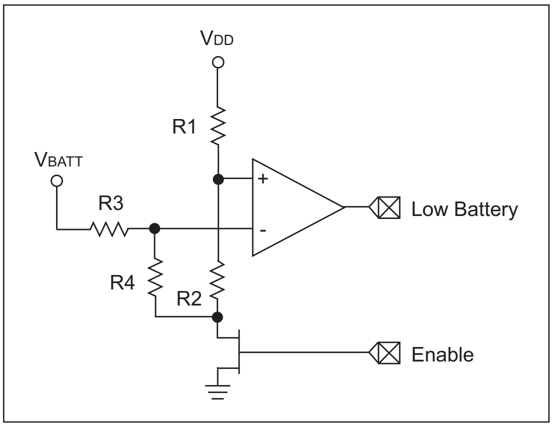
Equation 21-1

$$V_{IN} = V_{REF} \frac{T2}{T1}$$

TIP #1 Low Battery Detection

When operating from a battery power supply, it is important for a circuit to be able to determine when the battery charge is insufficient for normal operation of the circuit. Typically, this is a comparator-based circuit similar to the Programmable Low Voltage Detect (PLVD) peripheral. If the PLVD peripheral is not available in the microcontroller, a similar circuit can be constructed using a comparator and a few external components (see Figure 1-1 and Figure 1-2). The circuit in Figure 1-1 assumes that the microcontroller is operating from a regulated supply voltage. The circuit in Figure 1-2 assumes that the microcontroller supply is unregulated.

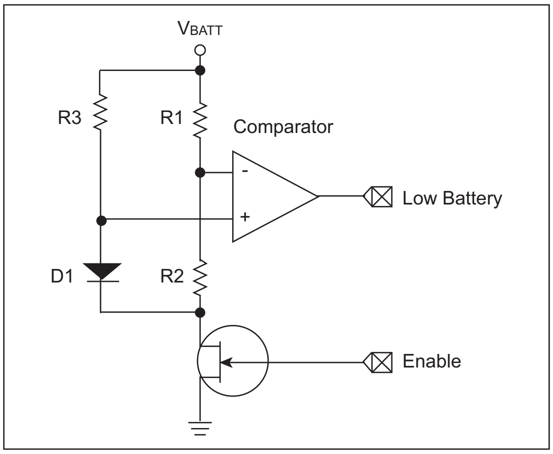
Figure 1-1: Regulated Supply



The comparator will trip when the battery voltage, V_{BATT} = 5.7V: R1 = 33k, R2 = 10k, R3 = 39k, R4 = 10k, V_{DD} = 5V.

In Figure 1-1, resistors R1 and R2 are chosen to place the voltage at the non-inverting input at approximately 25% of V_{DD}. R3 and R4 are chosen to set the inverting input voltage equal to the non-inverting input voltage when the battery voltage is equal to the minimum operating voltage for the system.

Figure 1-2: Unregulated Supply



Comparator will trip when V_{BATT} = 3V: R1 = 33k, R2 = 10k and R3 = 470Ω.

In Figure 1-2, resistor R3 is chosen to bias diode D1 above its forward voltage when V_{BATT} is equal to the minimum battery voltage for the system. Resistors R1 and R2 are chosen to set the inverting input voltage equal to the forward voltage of D1.

TIP #2 Faster Code for Detecting Change

When using a comparator to monitor a sensor, it is often just as important to know when a change occurs as it is to know what the change is. To detect a change in the output of a comparator, the traditional method has been to store a copy of the output and periodically compare the held value to the actual output to determine the change. An example of this type of routine is shown below.

Example 2-1

```
Test
   MOVF  hold,w     ;get old Cout
   XORWF CMCON,w    ;compare to new Cout
   ANDLW COUTMASK
   BTFSC STATUS,Z
   RETLW 0          ;if = return "no change"
   MOVF  CMCON,w    ;if not =, get new Cout
   ANDLW COUTMASK   ;remove all other bits
   MOVWF hold       ;store in holding var.
   IORLW CHNGBIT    ;add change flag
   RETURN
```

This routine requires 5 instructions for each test, 9 instructions if a change occurs, and 1 RAM location for storage of the old output state.

A faster method for microcontrollers with a single comparator is to use the comparator interrupt flag to determine when a change has occurred.

Example 2-2

```
Test
   BTFSS PIR1,CMIF  ;test comparator flag
   RETLW 0          ;if clear, return a 0
   BTFSS CMCON,COUT ;test Cout
   RETLW CHNGBIT    ;if clear return
                    ;CHNGFLAG
   RETLW COUTMASK + CHNGBIT;if set,
                    ;return both
```

This routine requires 2 instructions for each test, 3 instructions if a change occurs, and no RAM storage.

If the interrupt flag can not be used, or if two comparators share an interrupt flag, an alternate method that uses the comparator output polarity bit can be used.

Example 2-3

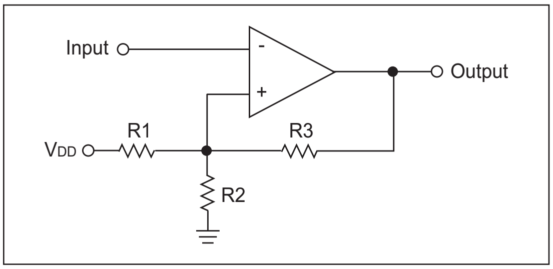
```
Test
   BTFSS CMCON,COUT ;test Cout
   RETLW 0          ;if clear, return 0
   MOVLW CINVBIT    ;if set, invert Cout
   XORWF CMCON,f    ;forces Cout to 0
   BTFSS CMCON,CINV ;test Cout polarity
   RETLW CHNGFLAG   ;if clear, return
                    ;CHNGFLAG
   RETLW COUTMASK + CHNGFLAG;if set,
                    ;return both
```

This routine requires 2 instructions for each test, 5 instructions if a change occurs, and no GPR storage.

TIP #3 Hysteresis

When the voltages on a comparator's input are nearly equal, external noise and switching noise from inside the microcontroller can cause the comparator output to oscillate or "chatter." To prevent chatter, some of the comparator output voltage is fed back to the non-inverting input of the comparator to form hysteresis (see Figure 3-1). Hysteresis moves the comparator threshold up when the input is below the threshold, and down when the input is above the threshold. The result is that the input must overshoot the threshold to cause a change in the comparator output. If the overshoot is greater than the noise present on the input, the comparator output will not chatter.

Figure 3-1: Comparator with Hysteresis



To calculate the resistor values required, first determine the high and low threshold values which will prevent chatter (V_{TH} and V_{TL}). Using V_{TH} and V_{TL}, the average threshold voltage can be calculated using the equation.

Equation 3-1

$$V_{AVG} = \frac{V_{DD} * V_{TL}}{V_{DD} - V_{TH} + V_{TL}}$$

Next, choose resistor values that satisfy Equation 3-2 and calculate the equivalent resistance using Equation 3-3.

Note: A continuous current will flow through R1 and R2. To limit the power dissipation in R1 and R2 the total resistance of R1 and R2 should be at least 1k. The total resistance of R1 and R2 should also be kept below 10K to keep the size of R3 small. Large values for R3, 100k-10 M , can produce voltage offsets at the non-inverting input due to the comparator's input bias current.

Equation 3-2

$$V_{AVG} = \frac{V_{DD} * R2}{R1 + R2}$$

Equation 3-3

$$R_{EQ} = \frac{R1 * R2}{R1 + R2}$$

Then, determine the feedback divider ratio D_R, using Equation 3-4.

Equation 3-4

$$D_R = \frac{(V_{TH} - V_{TL})}{V_{DD}}$$

Finally, calculate the feedback resistor R3 using Equation 3-5.

Equation 3-5

$$R3 = R_{EQ} \left[\left(\frac{1}{D_R} \right) - 1 \right]$$

Example:

- A V_{DD} = 5.0V, V_H = 3.0V and V_L = 2.5V
- V_{AVG} = 2.77V
- R = 8.2k and R2 = 10k, gives a V_{AVG} = 2.75V
- R_{EQ} = 4.5k
- D_R = .1
- R3 = 39k (40.5 calculated)
- V_{HACT} = 2.98V
- V_{LACT} = 2.46V

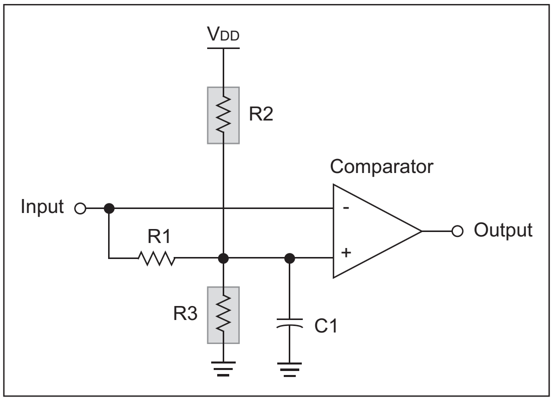
TIP #6 Data Slicer

In both wired and wireless data transmission, the data signal may be subject to DC offset shifts due to temperature shifts, ground currents or other factors in the system. When this happens, using a simple level comparison to recover the data is not possible because the DC offset may exceed the peak-to-peak amplitude of the signal. The circuit typically used to recover the signal in this situation is a data slicer.

The data slicer shown in Figure 6-1 operates by comparing the incoming signal with a sliding reference derived from the average DC value of the incoming signal. The DC average value is found using a simple RC low-pass filter (R1 and C1). The corner frequency of the RC filter should be high enough to ignore the shifts in the DC level while low enough to pass the data being transferred.

Resistors R2 and R3 are optional. They provide a slight bias to the reference, either high or low, to give a preference to the state of the output when no data is being received. R2 will bias the output low and R3 will bias the output high. Only one resistor should be used at a time, and its value should be at least 50 to 100 times larger than R1.

Figure 6-1: Data Slicer



Example:

Data rate of 10 kbits/second. A low pass filter frequency of 500 Hz: R1 = 10k, C1 = 33 μF. R2 or R3 should be 500k to 1 MB.

TIP #20 Logic: Set/Reset Flip Flop

This tip shows the use of the comparator to implement a Set/Reset Flip Flop.

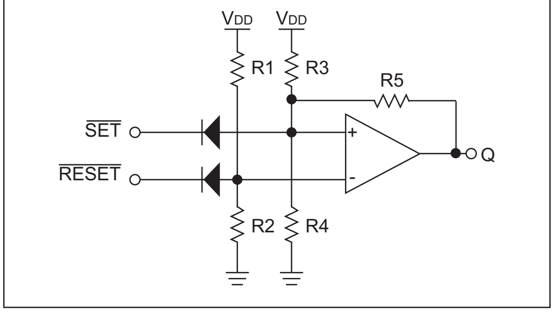
The inverting and non-inverting inputs are biases at $V_{DD}/2$ by resistors R1 through R4. The non-inverting input also receives positive feedback from the output through R5. The common bias voltages and the positive feedback configure the comparator as a bistable latch. If the output Q is high, the non-inverting input is also pulled high, which reinforces the high output. If Q is low, the non-inverting input is also pulled low, which reinforces the low output. To change state, the appropriate input must be pulled low to overcome the positive feedback. The diodes prevent a positive state on either input from pulling the bias of either input above $V_{DD}/2$.

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 Set and Reset must be driven near ground for the circuit to operate properly.
2. The combination of R1/R2 and R3/R4 will draw current constantly, so they must be kept large to minimize current draw.
3. R1 through R4 must be equal for a $V_{DD}/2$ trip level.
4. R5 must be greater or equal to R3.
5. R1 through R4 will react with the input capacitance of the comparator, so larger values will limit the minimum input pulse width.

Figure 20-1: Set/Reset Flip Flop



Example:

- Diodes = 1N4148
- R1 = R2 = R3 = R4 = 10k
- R5 = 10k

TIP #2 Brushless DC Motor Drive Circuits

A Brushless DC motor is a good example of simplified hardware increasing the control complexity. The motor cannot commutate the windings (switch the current flow), so the control circuit and software must control the current flow correctly to keep the motor turning smoothly. The circuit is a simple half-bridge on each of the three motor windings.

There are two basic commutation methods for Brushless DC motors; sensored and sensorless. Because it is critical to know the position of the motor so the correct winding can be energized, some method of detecting the rotor position is required. A motor with sensors will directly report the current position to the controller. Driving a sensored motor requires a look-up table. The current sensor position directly correlates to a commutation pattern for the bridge circuits.

Without sensors, another property of the motor must be sensed to find the position. A popular method for sensorless applications is to measure the back EMF voltage that is naturally generated by the motor magnets and windings. The induced voltage in the un-driven winding can be sensed and used to determine the current speed of the motor. Then, the next commutation pattern can be determined by a time delay from the previous pattern.

Sensorless motors are lower cost due to the lack of the sensors, but they are more complicated to drive. A sensorless motor performs very well in applications that don’t require the motor to start and stop. A sensor motor would be a better choice in applications that must periodically stop the motor.

Figure 2-1: 3 Phase Brushless DC Motor Control

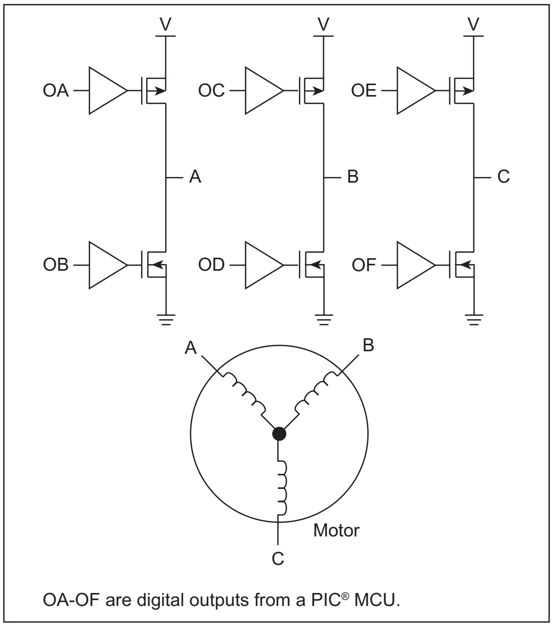
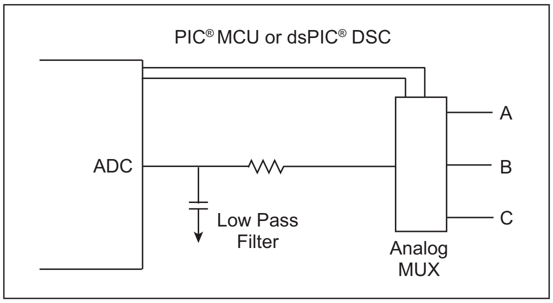


Figure 2-2: Back EMF Sensing (Sensorless Motor)



TIP #1 Typical Ordering Considerations and Procedures for Custom Liquid Displays

1. Consider what useful information needs to be displayed on the custom LCD and the combination of alphanumeric and custom icons that will be necessary.
2. Understand the environment in which the LCD will be required to operate. Operating voltage and temperature can heavily influence the contrast of the LCD and potentially limit the type of LCD that can be used.
3. Determine the number of segments necessary to achieve the desired display on the LCD and reference the PIC Microcontroller LCD matrix for the appropriate LCD PIC microcontroller.
4. Create a sketch/mechanical print and written description of the custom LCD and understand the pinout of the LCD. (Pinout definition is best left to the glass manufacturer due to the constraints of routing the common and segment electrodes in two dimensions.)
5. Send the proposed LCD sketch and description for a written quotation to at least 3 vendors to determine pricing, scheduling and quality concerns.
   a) Take into account total NRE cost, price per unit, as well as any setup fees.
   b) Allow a minimum of two weeks for formal mechanical drawings and pin assignments and revised counter drawings.
6. Request a minimal initial prototype LCD build to ensure proper LCD development and ensure proper functionality within the target application.
   a) Allow typically 4-6 weeks for initial LCD prototype delivery upon final approval of mechanical drawings and pin assignments.
7. Upon receipt of prototype LCD, confirm functionality before giving final approval and beginning production of LCD.

Note: Be sure to maintain good records by keeping copies of all materials transferred between both parties, such as initial sketches, drawings, pinouts, etc.

TIP #2 LCD PIC® MCU Segment/Pixel Table

Table 2-1: Segment Matrix Table

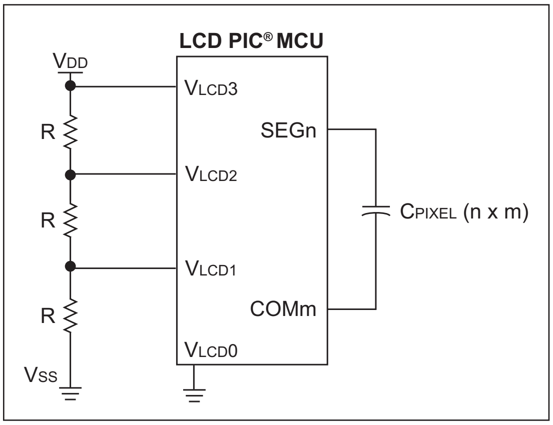
Multiplex Commons	Maximum Number of Segments/Pixels					Bias
	PIC16F913/916	PIC16F914/917	PIC16F946	PIC18F6X90 (PIC18F6XJ90)	PIC18F8X90 (PIC18F8XJ90)	
Static (COM0)	15	24	42	32/ (33)	48	Static
1/2 (COM1: COM0)	30	48	84	64/ (66)	96	1/2 or 1/3
1/3 (COM2: COM0)	45	72	126	96/ (99)	144	1/2 or 1/3
1/4 (COM3: COM0)	60	96	168	128/ (132)	192	1/3

This Segment Matrix table shows that Microchip's 80-pin LCD devices can drive up to 4 commons and 48 segments (192 pixels), 64-pin devices can drive up to 33 segments (132 pixels), 40/44 pin devices can drive up to 24 segments (96 pixels) and 28-pin devices can drive 15 segments (60 segments).

TIP #3 Resistor Ladder for Low Current

Bias voltages are generated by using an external resistor ladder. Since the resistor ladder is connected between V_{DD} and V_{SS}, there will be current flow through the resistor ladder in inverse proportion to the resistance. In other words, the higher the resistance, the less current will flow through the resistor ladder. If we use 10K resistors and V_{DD} = 5V, the resistor ladder will continuously draw 166 µA. That is a lot of current for some battery-powered applications.

Figure 3-1: Resistor Ladder

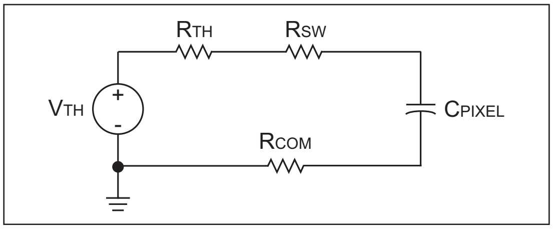


How do we maximize the resistance without adversely effecting the quality of the display? Some basic circuit analysis helps us determine how much we can increase the size of the resistors in the ladder.

The LCD module is basically an analog multiplexer that alternately connects the LCD voltages to the various segment and common pins that connect across the LCD pixels. The LCD pixels can be modeled as a capacitor. Each tap point on the resistor ladder can be modeled as a Thevenin equivalent circuit. The Thevenin resistance is 0 for $V_{LCD}3$ and $V_{LCD}0$, so we look at the two cases where it is non-zero, $V_{LCD}2$ and $V_{LCD}1$.

The circuit can be simplified as shown in Figure 3-2. R_{SW} is the resistance of the segment multiplex switch; R_{COM} is the resistance of the common multiplex switch.

Figure 3-2: Simplified LCD Circuit

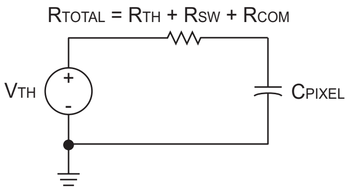


The Thevenin voltage is equal to either 2/3 V_{DD}, or 1/3 V_{DD}, for the cases where the Thevenin resistance is non-zero. The Thevenin resistance is equal to the parallel resistance of the upper and lower parts of the resistor ladder.

Figure 3-3: LCD Circuit Resistance Estimate

$R_{TH} = (2R * R)/(2R + R)$
$R_{TH} = 2R^2/3R$
$R_{TH} = 2R/3$

$R_{SW} = 4.7K$
$R_{COM} = 0.4K$



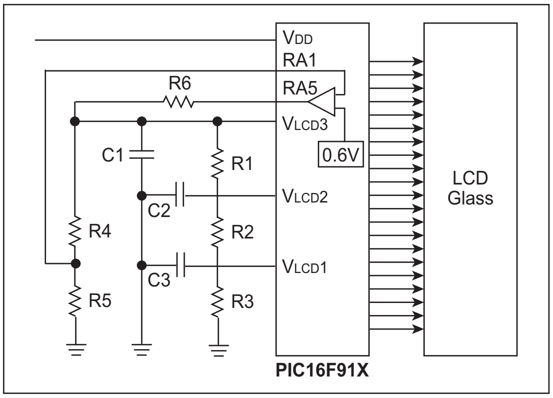
As you can see, we can model the drive of a single pixel as an RC circuit, where the voltage switches from 0V to $V_{LCD}2$, for example. For LCD PIC microcontrollers, we can estimate the resistance of the segment and common switching circuits as about 4.7K and 0.4K, respectively.

We can see that the time for the voltage across the pixel to change from 0 to V_{TH} will depend on the capacitance of the pixel and the total resistance, of which the resistor ladder Thevenin resistance forms the most significant part.

TIP #4: Contrast Control with a Buck Regulator

Contrast control in any of the LCD PIC MCUs is accomplished by controlling the voltages applied to the VLCD voltage inputs. The simplest contrast voltage generator is to place a resistor divider across the three pins. This circuit is shown in the data sheet. The resistor ladder method is good for many applications, but the resistor ladder does not work in an application where the contrast must remain constant over a range of VDDS. The solution is to use a voltage regulator. The voltage regulator can be external to the device, or it can be built using a comparator internal to the LCD PIC microcontroller.

Figure 4-1: Voltage Generator with Resistor Divider



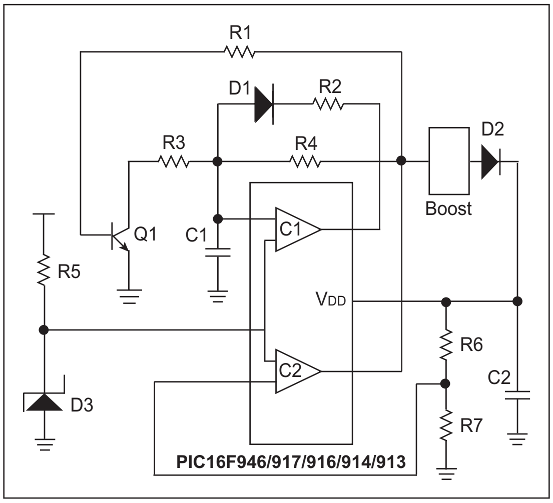
The PIC16F946/917/916/914/913 devices have a special Comparator mode that provides a fixed 0.6V reference. The circuit shown in Figure 4-1 makes use of this reference to provide a regulated contrast voltage. In this circuit, R1, R2 and R3 provide the contrast control voltages. The voltage on VLCD3 is compared to the internal voltage reference by dividing the voltage at VLCD3 at R4 and R5 and applying the reduced voltage to the internal comparator. When the voltage at VLCD3 is close to the desired voltage, the output of the comparator will begin to oscillate. The oscillations are filtered into a DC voltage by R6 and C1. C2 and C3 are simply small bypass capacitors to ensure that the voltages at VLCD1 and VLCD2 are steady.

TIP #5: Contrast Control Using a Boost Regulator

In LCD Tip #4, a buck converter was created using a comparator. This circuit works great when VDD is greater than the LCD voltage. The PIC microcontroller can operate all the way down to 2.0V, whereas most low-voltage LCD glass only operates down to 3V. In a battery application, it is important to stay operational as long as possible. Therefore, a boost converter is required to boost 2.0V up to 3.0V for the LCD.

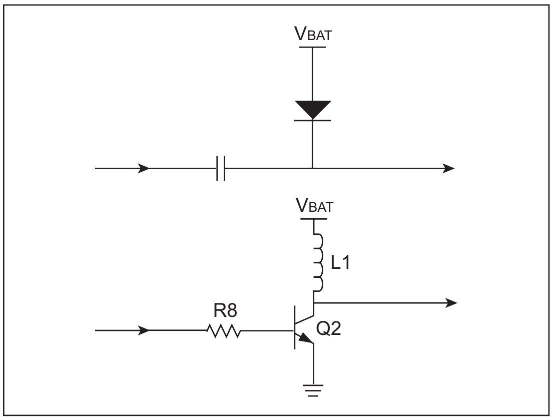
The figure below shows one circuit for doing this.

Figure 5-1: Boost Converter



In this circuit, both comparators are used. The voltage setpoint is determined by the value of Zenier diode D3 and the voltage at R6:R7. The rest of the circuit creates a simple multivibrator to stimulate a boost circuit. The boost circuit can be inductor or capacitor-based. When the output voltage is too low, the multivibrator oscillates and causes charge to build up in C2. As the voltage at C2 increases, the multivibrator will begin to operate sporadically to maintain the desired voltage at C2.

Figure 5-2: Two Types of Boost Converter

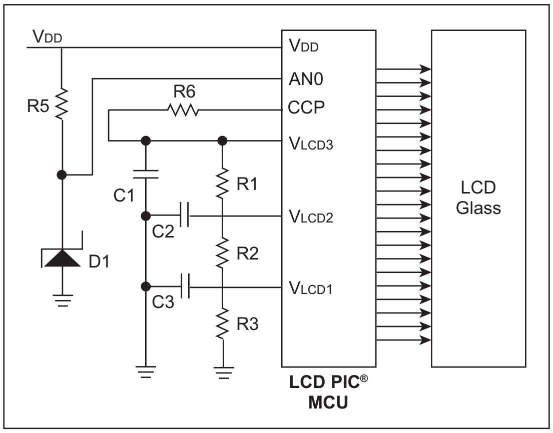


The two methods of producing a boost converter are shown above. The first circuit is simply a switched capacitor type circuit. The second circuit is a standard inductor boost circuit. These circuits work by raising V_{DD}. This allows the voltage at V_{LCD} to exceed V_{DD}.

TIP #6: Software Controlled Contrast with PWM for LCD Contrast Control

In the previous contrast control circuits, the voltage output was set by a fixed reference. In some cases, the contrast must be variable to account for different operating conditions. The CCP module, available in the LCD controller devices, allows a PWM signal to be used for contrast control. In Figure 6-1, you see the buck contrast circuit modified by connecting the input to RA6 to a CCP pin. The resistor divider created by R4 and R5 in the previous design are no longer required. An input to the ADC is used to provide feedback but this can be considered optional. If the ADC feedback is used, notice that it is used to monitor the V_{DD} supply. The PWM will then be used to compensate for variations in the supply voltage.

Figure 6-1: Software Controlled Voltage Generator



TIP #8 In-Circuit Debug (ICD)

There are two potential issues with using the ICD to debug LCD applications. First, the LCD controller can freeze while the device is Halted. Second, the ICD pins are shared with segments on the PIC16F946/917/916/914/913 MCUs.

When debugging, the device is Halted at breakpoints and by the user pressing the pause button. If the ICD is configured to Halt the peripherals with the device, the LCD controller will Halt and apply DC voltages to the LCD glass. Over time, these DC levels can cause damage to the glass; however, for most debugging situations, this will not be a consideration. The PIC18F LCD MCUs have a feature that allows the LCD module to continue operating while the device has been Halted during debugging. This is useful for checking the image of the display while the device is Halted and for preventing glass damage if the device will be Halted for a long period of time.

The PIC16F946/917/916/914/913 multiplex the ICSP™ and ICD pins onto pins shared with LCD segments 6 and 7. If an LCD is attached to these pins, the device can be debugged with ICD; however, all the segments driven by those two pins will flicker and be uncontrolled. As soon as debugging is finished and the device is programmed with Debug mode disabled, these segments will be controlled correctly.

TIP #9 LCD in Sleep Mode

If you have a power-sensitive application that must display data continuously, the LCD PIC microcontroller can be put to Sleep while the LCD driver module continues to drive the display.

To operate the LCD in Sleep, only two steps are required. First, a time source other than the main oscillator must be selected as the LCD clock source, because during Sleep, the main oscillator is Halted. Options are shown for the various LCD PIC MCUs.

Table 9-1: Options for LCD in Sleep Mode

Part	LCD Clock Source	Use in Sleep?
PIC16C925/926	FOSC/256	No
	T1OSC	Yes
	Internal RC Oscillator	Yes
PIC16F946/917/916/914/913	FOSC/8192	No
	T1OSC/32	Yes
	LFINTOSC/32	Yes
PIC18F6X90	(FOSC/4)/8192	No
PIC18F8X90	T1OSC	Yes
PIC18F6XJ90	INTRC/32	Yes
PIC18F8XJ90		

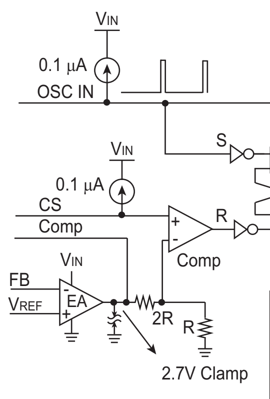
Second, the Sleep Enable bit (SLPEN) must be cleared. The LCD will then continue to display data while the part is in Sleep. It’s that easy!

When should you select the internal RC oscillator (or LFINTOSC) over the Timer1 oscillator? It depends on whether your application is time-sensitive enough to require the accuracy of a crystal on the Timer1 oscillator or not. If you have a timekeeping application, then you will probably have a 32 kHz crystal oscillator connected to Timer1.

Since Timer1 continues to operate during Sleep, there is no penalty in using Timer1 as the LCD clock source. If you don’t need to use an external oscillator on Timer1, then the internal RC oscillator (INTRC or LFINTOSC) is more than sufficient to use as the clock source for the LCD and it requires no external components.

TIP #6 Current Limiting Using the MCP1630

Figure 6-1: MCP1630 High-Speed PWM



t	t	e
		—
0	0	$\overline{Qn}$
0	1	1
1	0	0
1	1	1

Note 1: During overtemperature, V_{EXT} driver is high-impedance.

The block diagram for the MCP1630 high-speed PWM driver is shown in Figure 6-1. One of the features of the MCP1630 is the ability to perform current limiting. As shown in the bottom left corner of the diagram, the output of the Error Amplifier (EA) is limited by a 2.7V clamp. Therefore, regardless of the actual error, the input to the negative terminal of the comparator (labeled Comp) is limited to 2.7V ÷ 3 or 0.9V.

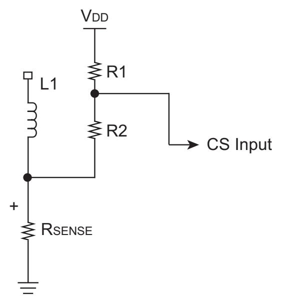
It is possible to implement the current limiting by using a single sense resistor. In this case, the maximum current would be given by Equation 6-1.

Equation 6-1

$$I_{MAX} = (0.9V) / R_{SENSE}$$

For high current applications, this method may be acceptable. When lower current limits are required, the size of the sense resistor, R_{SENSE}, must be increased. This will cause additional power dissipation. An alternative method for lower current limits is shown in Figure 6-2.

Figure 6-2: Low Current Limits



In this case, the Current Sense (CS) input of the MCP1630 is biased upward using the R1/R2 resistor divider. The equations for the new current limit are shown in Equation 6-2.

Equation 6-2

$$0.9V = \frac{(V_{DD} - I_{MAX} \bullet R_{SENSE}) \bullet R2}{R1 + R2}$$

Equation 6-2 can be solved to determine the values of R1 and R2 that provide the desired current limit.

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.

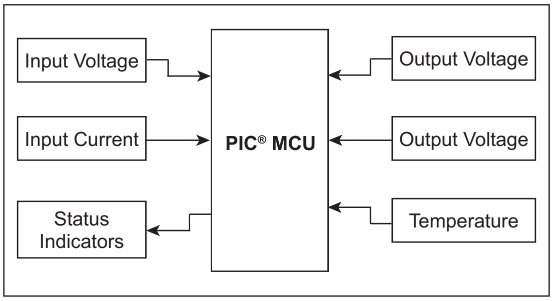
The following application notes are related to PID control algorithms and all include example software:

- 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 #5 3.3V → 5V Direct Connect

The simplest and most desired way to connect a 3.3V output to a 5V input is by a direct connection. This can be done only if the following 2 requirements are met:

- The V_{OH} of the 3.3V output is greater than the V_{IH} of the 5V input
- The V_{OL} of the 3.3V output is less than the V_{IL} of the 5V input

An example of when this technique can be used is interfacing a 3.3V LVCMOS output to a 5V TTL input. From the values given in Table 4-1, it can clearly be seen that both of these requirements are met.

3.3V LVCMOS V_{OH} of 3.0 volts is greater than 5V TTL V_{IH} of 2.0 volts, and

3.3V LVCMOS V_{OL} of 0.5 volts is less than 5V TTL V_{IL} of 0.8 volts.

When both of these requirements are not met, some additional circuitry will be needed to interface the two parts. See Tips 6, 7, 8 and 13 for possible solutions.

TIP #6 3.3V → 5V Using a MOSFET Translator

In order to drive any 5V input that has a higher V_{IH} than the V_{OH} of a 3.3V CMOS part, some additional circuitry is needed. A low-cost two component solution is shown in Figure 6-1.

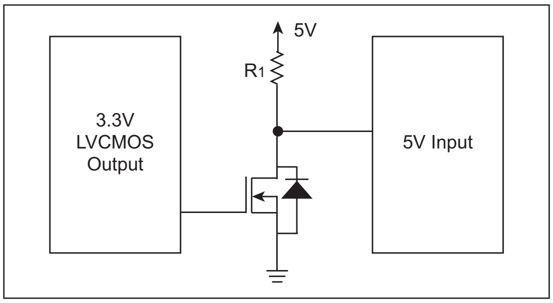
When selecting the value for R1, there are two parameters that need to be considered; the switching speed of the input and the current consumption through R1. When switching the input from a ‘0’ to a ‘1’, you will have to account for the time the input takes to rise because of the RC time constant formed by R1, and the input capacitance of the 5V input plus any stray capacitance on the board. The speed at which you can switch the input is given by the following equation:

Equation 6-1

$$T_{SW} = 3 \times R_1 \times (C_{IN} + C_S)$$

Since the input and stray capacitance of the board are fixed, the only way to speed up the switching of the input is to lower the resistance of R1. The trade-off of lowering the resistance of R1 to get faster switching times is the increase in current draw when the 5V input remains low. The switching to a ‘0’ will typically be much faster than switching to a ‘1’ because the ON resistance of the N-channel MOSFET will be much smaller than R1. Also, when selecting the N-channel FET, select a FET that has a lower V_{GS} threshold voltage than the V_{OH} of 3.3V output.

Figure 6-1: MOSFET Translator


5V
R1
3.3V LVCMOS Output
5V Input