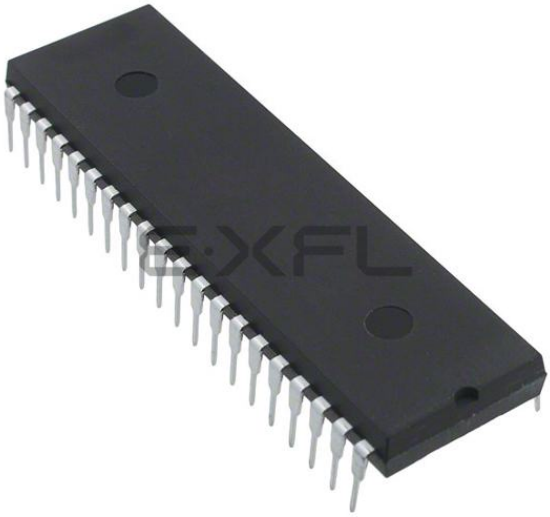


Welcome to E-XFL.COM

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.

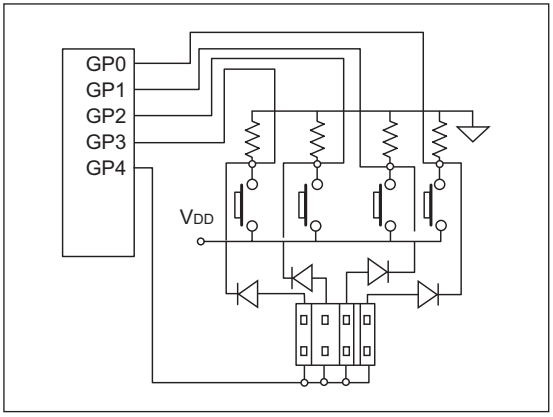
Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	33
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 ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	40-DIP (0.600", 15.24mm)
Supplier Device Package	40-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16c65b-20i-p

TIP #9 Decode Keys and ID Settings

Buttons and jumpers can share I/O’s by using another I/O to select which one is read. Both buttons and jumpers are tied to a shared pull-down resistor. Therefore, they will read as ‘0’ unless a button is pressed or a jumper is connected. Each input (GP3/2/1/0) shares a jumper and a button. To read the jumper settings, set GP4 to output high and each connected jumper will read as ‘1’ on its assigned I/O or ‘0’ if it’s not connected. With GP4 output low, a pressed button will be read as ‘1’ on its assigned I/O and ‘0’ otherwise.

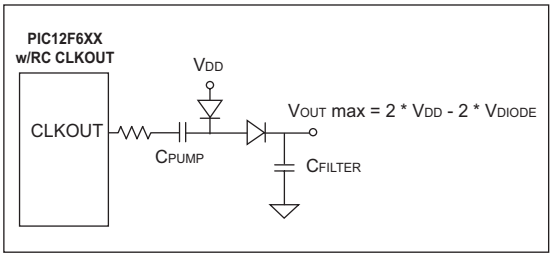
Figure 9-1



- When GP4 = 1 and no keys are pressed, read ID setting
- When GP4 = 0, read the switch buttons

TIP #10 Generating High Voltages

Figure 10-1



Voltages greater than V_{DD} can be generated using a toggling I/O. PIC MCUs CLKOUT/OSC2 pin toggles at one quarter the frequency of OSC1 when in external RC oscillator mode. When OSC2 is low, the V_{DD} diode is forward biased and conducts current, thereby charging C_{PUMP}. After OSC2 is high, the other diode is forward biased, moving the charge to C_{FILTER}. The result is a charge equal to twice the V_{DD} minus two diode drops. This can be used with a PWM, a toggling I/O or other toggling pin.

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.

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.

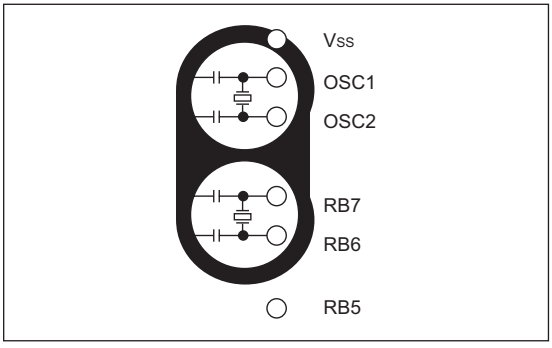
Some nanoWatt devices and all nanoWatt XLP devices can also use the Timer1/SOSC oscillator as the system clock source in place of the main oscillator on the OSC1/OSC2 pins. By reducing execution speed, total current consumption can be reduced.

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

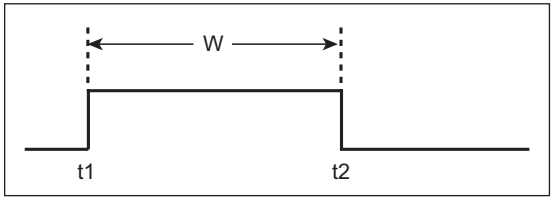
The Low Voltage Detect (LVD) interrupt present in many PIC MCUs is critical in battery based systems. It is necessary for two reasons. First, many devices cannot run full speed at the minimum operating voltage. In this case, the LVD interrupt indicates when the battery voltage is dropping so that the 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 #3 Measuring Pulse Width

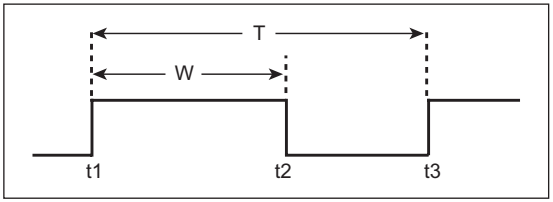
Figure 3-1: Pulse Width



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.

COMPARE TIPS ‘N TRICKS

In Compare mode, the 16-bit CCPRx register value is constantly compared against the TMR1 register pair values. When a match occurs, the CCPx pin is:

- Driven high
- Driven low
- Remains unchanged, or
- Toggles based on the module’s configuration

The action on the pin is determined by control bits CCPxM3:CCPxM0 (CCPxCON<3:0>). A CCP interrupt is generated when a match occurs.

Special Event Trigger

Timer1 is normally not cleared during a CCP interrupt when the CCP module is configured in Compare mode. The only exception to this is when the CCP module is configured in Special Event Trigger mode. In this mode, when Timer1 and CCPRx are equal, the CCPx interrupt is generated, Timer1 is cleared, and an A/D conversion is started (if the A/D module is enabled.)

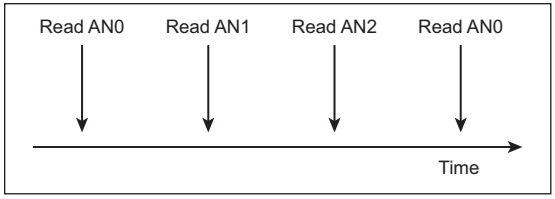
“Why Would I Use Compare Mode?”

Compare mode works much like the timer function on a stopwatch. In the case of a stopwatch, a predetermined time is loaded into the watch and it counts down from that time until zero is reached.

Compare works in the same way with one exception – it counts from zero to the predetermined time. This mode is useful for generating specific actions at precise intervals. A timer could be used to perform the same functionality, however, it would mean preloading the timer each time. Compare mode also has the added benefit of automatically altering the state of the CCPx pin based on the way the module is set up.

TIP #11 Sequential ADC Reader

Figure 11-1: Timeline



Trigger Special Event mode (a sub-mode in Compare mode) generates a periodic interrupt in addition to automatically starting an A/D conversion when Timer1 matches CCPRxL and CCPRxH. The following example problem demonstrates how to sequentially read the A/D channels at a periodic interval.

Example

Given the PIC16F684 running on its 8 MHz internal oscillator, configure the microcontroller to sequentially read analog pins AN0, AN1 and AN2 at 30 ms intervals.

Step #1: Determine Timer1 Prescaler

a) Timer1 overflows at: Tosc*4*65536* prescaler.

b) For a prescaler of 1:1, the Timer1 overflow occurs in 32.8 ms.

c) This is greater than 30 ms, so a prescaler of 1 is adequate.

Step #2: Calculate CCPR1 (CCPR1L and CCPR1H)

a) CCPR1 = Interval Time/(Tosc*4*prescaler) = 0.030/(125 ns*4*1) = 6000 = 0xEA60

b) Therefore, CCPR1L = 0x60, and CCPR1H = 0xEA

Step #3: Configuring CCP1CON

The ECCP module should be configured in Trigger Special Event mode. This mode generates an interrupt when Timer1 equals the value specified in CCPR1. Timer1 is automatically cleared and the GO bit in ADCON0 is automatically set. For this mode, CCP1CON = '`b00001011`'.

Step #4: Add Interrupt Service Routine Logic

When the ECCP interrupt is generated, select the next A/D pin for reading by altering the ADCON0 register.

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 #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.

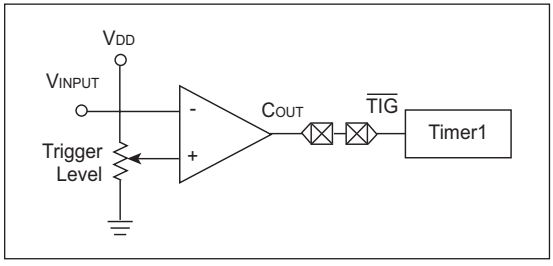
TIP #4 Pulse Width Measurement

To measure the high or low pulse width of an incoming analog signal, the comparator can be combined with Timer1 and the Timer1 Gate input option (see Figure 4-1). Timer1 Gate acts as a count enable for Timer1. If the input is low, Timer1 will count. If the $\overline{T1G}$ input is high, Timer1 does not count. Combining $\overline{T1G}$ with the comparator allows the designer to measure the time between a high-to-low output change and a low-to-high output change.

To make a measurement between a low-to-high and a high-to-low transition, the only change required is to set the CINV bit in the comparator CMCON register which inverts the comparator output.

Because the output of the comparator can change asynchronously with the Timer1 clock, only comparators with the ability to synchronize their output with the Timer1 clock should be used and their C2SYNC bits should be set.

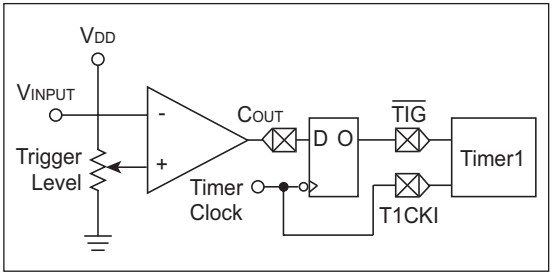
Figure 4-1: Comparator with Timer1 and $\overline{T1G}$



If the on-chip comparator does not have the ability to synchronize its output to the Timer1 clock, the output can be synchronized externally using a discrete D flip-flop (see Figure 4-2).

Note: The flip-flop must be falling edge triggered to prevent a race condition.

Figure 4-2: Externally Synchronized Comparator

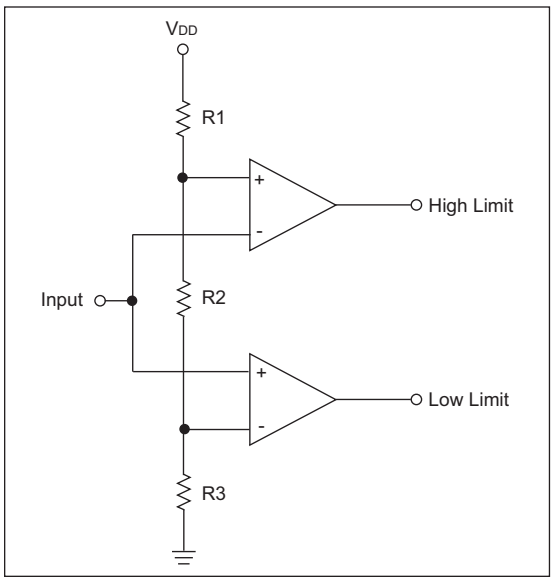


TIP #5 Window Comparison

When monitoring an external sensor, it is often convenient to be able to determine when the signal has moved outside a pre-established safe operating range of values or window of operation. This windowing provides the circuit with an alarm when the signal moves above or below safety limits, ignoring minor fluctuations inside the safe operating range.

To implement a window comparator, two voltage comparators and 3 resistors are required (see Figure 5-1).

Figure 5-1: Window Comparator



Resistors R1, R2 and R3 form a voltage divider which generates the high and low threshold voltages. The outputs HIGH LIMIT and LOW LIMIT are both active high, generating a logic one on the HIGH LIMIT output when the input voltage rises above the high threshold, and a logic one on the LOW LIMIT output when the input voltage falls below the low threshold.

To calculate values for R1, R2 and R3, find values that satisfy Equation 5-1 and Equation 5-2.

Note: A continuous current will flow through R1, R2 and R3. To limit the power dissipation in the resistors, the total resistance of R1, R2 and R3 should be at least 1k. The total resistance of R1, R2 and R3 should also be kept less than 1 M to prevent offset voltages due to the input bias currents of the comparator.

Equation 5-1

$$V_{TH\text{-}HI} = \frac{V_{DD} * (R_3 + R_2)}{R1 + R2 + R3}$$

Equation 5-2

$$V_{TH\text{-}LO} = \frac{V_{DD} * R_3}{R1 + R2 + R3}$$

Example:

- V_{DD} = 5.0V, V_{TH} = 2.5V, V_{TL} = 2.0V
- R1 = 12k, R2 = 2.7k, R3 = 10k
- V_{TH} (actual) = 2.57V, V_{TL} (actual) = 2.02V

Adding Hysteresis:

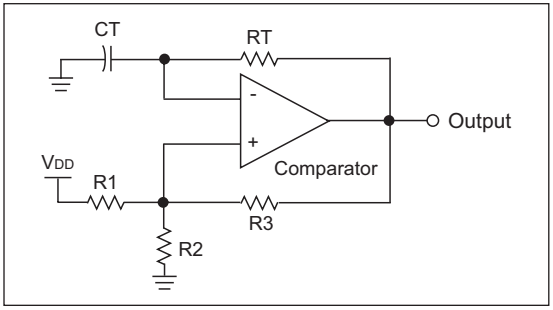
To add hysteresis to the HIGH LIMIT comparator, follow the procedure outlined in Tip #3. Use the series combination of R2 and R3 as the resistor R2 in Tip #3.

To add hysteresis to the LOW LIMIT comparator, choose a suitable value for Req, 1k to 10 k , and place it between the circuit input and the non-inverting input of the LOW LIMIT comparator. Then calculate the needed feedback resistor using Equation 3-4 and Equation 3-5.

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 μF for F_{OSC} = 480 Hz

TIP #6 Current Sensing

The torque of an electric motor can be monitored and controlled by keeping track of the current flowing through the motor. Torque is directly proportional to the current. Current can be sensed by measuring the voltage drop through a known value resistor or by measuring the magnetic field strength of a known value inductor. Current is generally sensed at one of two places, the supply side of the drive circuit (high side current sense) or the sink side of the drive circuit (low side current sense). Low side sensing is much simpler but the motor will no longer be grounded, causing a safety issue in some applications. High side current sensing generally requires a differential amplifier with a common mode voltage range within the voltage of the supply.

Figure 6-1: Resistive High Side Current Sensing

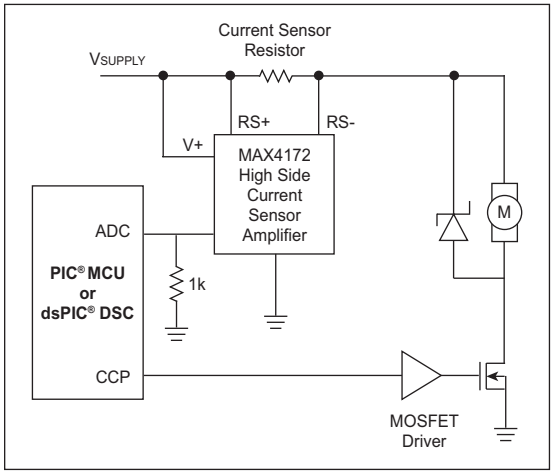
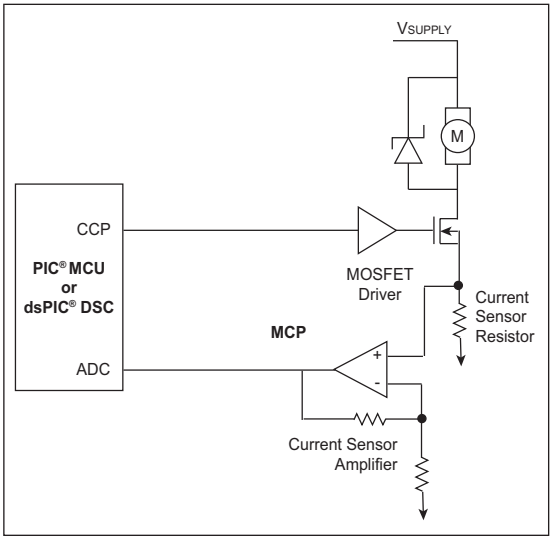
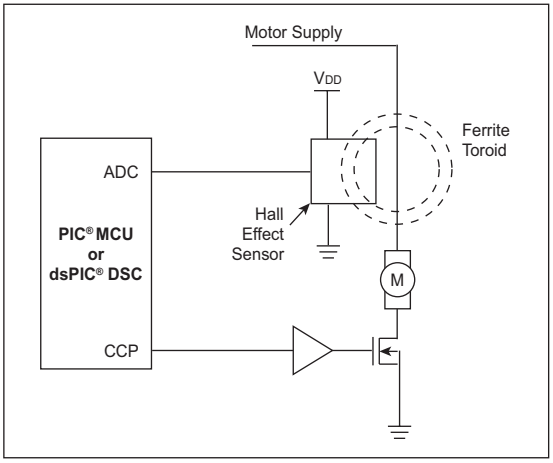


Figure 6-2: Resistive Low Side Current Sensing



Current measurement can also be accomplished using a Hall effect sensor to measure the magnetic field surrounding a current carrying wire. Naturally, this Hall effect sensor can be located on the high side or the low side of the load. The actual location of the sensor does not matter because the sensor does not rely upon the voltage on the wire. This is a non-intrusive method that can be used to measure motor current.

Figure 6-3: Magnetic Current Sensing



TIP #10 How to Update LCD Data Through Firmware

To update the LCD, the content of the LCDDATA registers is modified to turn on, or off, each pixel on the LCD display. The application firmware will usually modify buffer variables that are created to correspond with elements on the display, such as character positions, bar graph, battery display, etc.

When the application calls for a display update, the values stored in the buffer variables must be converted to the correct setting of the pixel bits, located in the LCDDATA registers.

For Type-A waveforms, the LCD Data registers may be written any time without ill effect. However, for Type-B waveforms, the LCD Data registers can only be written every other LCD frame in order to ensure that the two frames of the Type-B waveform are compliments of one another. Otherwise, a DC bias can be presented to the LCD.

The LCD Data registers should only be written when a write is allowed, which is indicated by the WA bit in the LCDCON register being set.

On the PIC16C926 parts, there is no WA bit. The writing of the pixel data can be coordinated on an LCD interrupt. The LCD interrupt is only generated when a multiplexed (not static) Type-B waveform is selected.

TIP #11 Blinking LCD

Information can be displayed in more than one way with an LCD panel. For example, how can the user's attention be drawn to a particular portion of the LCD panel? One way that does not require any additional segments is to create a blinking effect.

Look at a common clock application. The ":" between the hours and minutes is commonly made to blink once a second (on for half a second, off for half a second). This shows that the clock is counting in absence of the ticking sound or second hand that accompanies the usual analog face clock. It serves an important purpose of letting the user know that the clock is operating.

If there is a power outage, then it is common for the entire clock display to blink. This gives the user of the clock an immediate indication that the clock is no longer showing the correct time.

When the user sets the time, then blinking is commonly used to show that a new mode has been entered, such as blinking the hours to identify that the hours are being set, or blinking the minutes to show that the minutes are being set. In a simple clock, blinking is used for several different purposes. Without blinking effects, the common digital clock would not be nearly as user friendly.

The two digital I/O pins that are used are RB0 and RB5, but any two digital I/O pins could work. The two analog pins used are AN0 and AN1.

To read the keypad, follow the steps below:

1. First, make RB0 an output high and RB5 an input (to present a high impedance).
2. Perform two successive A/D conversions, first on AN0, then on AN1.
3. Save the conversion results to their respective variables; for example, `RB0_AN0_Result` and `RB0_AN1_Result`.
4. Next, make RB5 an output high and RB0 an input (to present a high impedance).
5. Perform two successive A/D conversions, first on AN0, then on AN1.
6. Save the conversion results to their respective variables; for example, `RB5_AN0_Result` and `RB5_AN1_Result`.
7. There are now 4 variables that represent a key press in each quadrant of the 4 x 4 keypad:
   - `RB0_AN0_Result` denotes key press of 1, 2, 4 or 5
   - `RB0_AN1_Result` denotes key press of 7, 8, A or 0
   - `RB5_AN0_Result` denotes key press of 3, C, 6 or D
   - `RB5_AN1_Result` denotes key press of 9, E, B or F
8. Finally, check each value against the matching column of Table 12-1. If it is within ±10% of a value, then it can be taken to indicate that the corresponding key has been pressed.

Table 12-1: Keypad Values

Value ±10%	RB0_AN0	RB0_AN1	RB5_AN0	RB5_AN1
<V_{DD}/10	–	–	–	–
V_{DD}/5.2	2	8	C	E
V_{DD}/4.2	1	7	3	9
V_{DD}/3	5	0	D	F
V_{DD}/2	4	A	6	B

9. This loop should be repeated about once every 20 ms or so.

Don't forget a debounce routine. For example, require the above steps (with 20 ms delay between) to return the same key value twice in a row for that key to be considered pressed. Also, require a no key press to be returned at least twice before looking for the next key press.

When keys within the same quadrant are pressed simultaneously, voltages other than the four valid levels shown in the table may be generated. These levels can either be ignored, or if you want to use simultaneous key presses to enable certain functions, you can add decoding for those levels as well.

TIP #3 A Tracking and Proportional Soft-Start of Two Power Supplies

Expanding on the previous tip, we can also use a PIC MCU to ensure that two voltages in a system rise together or rise proportionally to one another, as shown in Figure 3-1. This type of start-up is often used in applications with devices that require multiple voltages (such as I/O and core voltages).

Like the previous two, this tip is designed to control the shutdown pin of the SMPS controller and will only work with controllers that respond quickly to changes on the shutdown pin.

Figure 3-1: Timing Diagram

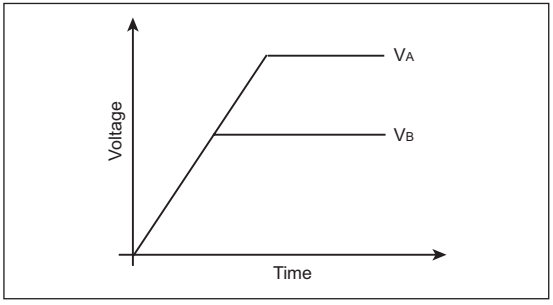
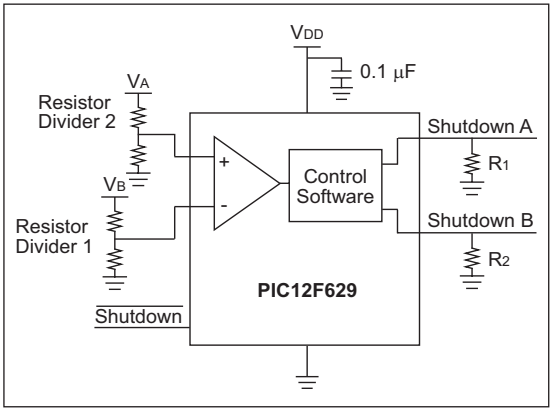


Figure 3-2: Example Schematic



The comparator of the PIC MCU is used to determine which voltage is higher and increases the on-time of the other output accordingly. The logic for the shutdown pins is as shown in Table 3-1.

Table 3-1: Shutdown Pin Logic

Case	Shutdown A	Shutdown B
$V_A > V_B$	Low	High
$V_B > V_A$	High	Low
V_B > Internal Reference	High	High

To determine if it has reached full voltage, V_B is compared to the internal voltage reference. If V_B is higher, both shutdown outputs are held high.

Resistor Divider 1 should be designed so that the potentiometer output is slightly higher than the comparator voltage reference when V_B is at full voltage.

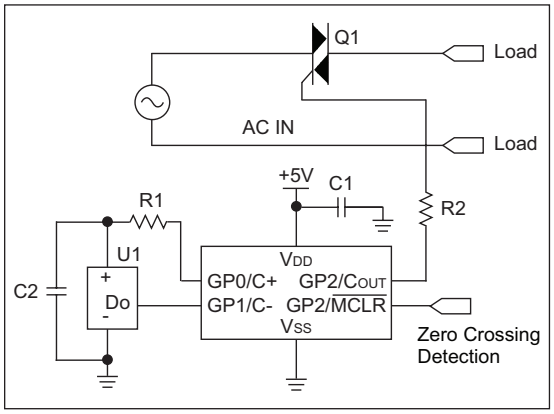
The ratio of resistors in Resistor Divider 2 can be varied to change the slope at which V_A rises.

Pull-down resistors ensure the power supplies will not operate unexpectedly when the PIC MCU is being reset.

TIP #9 An IR Remote Control Actuated AC Switch for Linear Power Supply Designs

Many line-powered applications (audio amplifiers, televisions, etc.) can be turned on and off using an infrared remote control. This requires that some components be energized to receive the remote signals even when the device is off. Low current PIC microcontrollers are best in this application. Figure 9-1 shows an example circuit layout.

Figure 9-1: PIC MCU Infrared Receiver Schematic



The PIC10F200 has several features that make it ideally suited for this type of application:

- Extremely low operating and standby current (350 µA operating, 0.1 µA when asleep)
- Input/Output pins with configurable pull-ups and reset-on-change capability
- High sink/source ability (±25 mA) allows driving external devices, such as the IR receiver, directly from the I/O pin
- Ability to use a low-cost resistive power supply
- Small form factor (SOT-23 packaging)

TB094, “*Dimming AC Incandescent Lamps Using A PIC10F200*” (DS91094) provides both software and hardware examples of an infrared controller.

TIP #20 Compensating Sensors Digitally

Many sensors and references tend to drift with temperature. For example, the MCP9700 specification states that its typical is ±0.5°C and its max error is ±4°C.

Figure 20-1: MCP9700 Accuracy

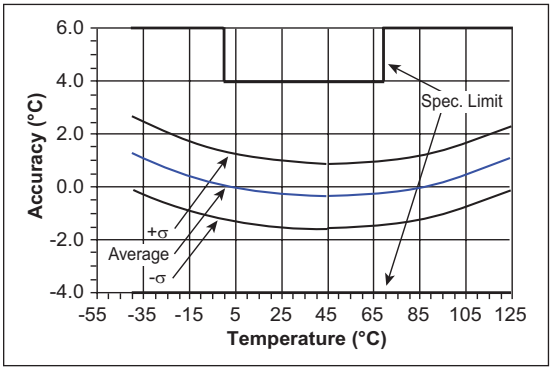


Figure 20-1 shows the accuracy of a 100 sample lot of MCP9700 temperature sensors. Despite the fact that the sensor’s error is nonlinear, a PIC microcontroller (MCU) can be used to compensate the sensor’s reading.

Polynomials can be fitted to the average error of the sensor. Each time a temperature reading is received, the PIC MCU can use the measured result and the error compensation polynomials to determine what the true temperature is.

Figure 20-2: MCP9700 Average Accuracy After Compensation

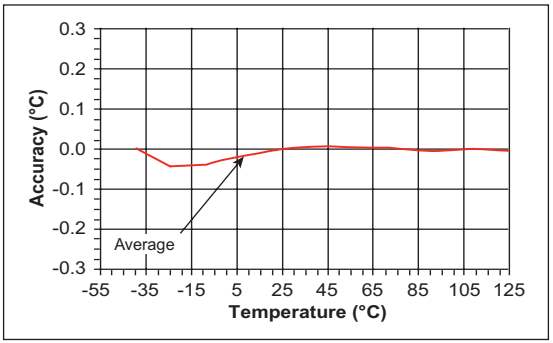


Figure 20-2 shows the average accuracy for the 100 sample lot of MCP9700 temperature sensors after compensation. The average error has been decreased over the full temperature range.

It is also possible to compensate for error from voltage references using this method.

For more information on compensating a temperature sensor digitally, refer to AN1001, “*IC Temperature Sensor Accuracy Compensation with a PIC Microcontroller*” (DS01001).

TIP #4 Powering 3.3V Systems From 5V Using Switching Regulators

A buck switching regulator, shown in Figure 4-1, is an inductor-based converter used to step-down an input voltage source to a lower magnitude output voltage. The regulation of the output is achieved by controlling the ON time of MOSFET Q1. Since the MOSFET is either in a lower or high resistive state (ON or OFF, respectively), a high source voltage can be converted to a lower output voltage very efficiently.

The relationship between the input and output voltage can be established by balancing the volt-time of the inductor during both states of Q1.

Equation 4-1

$$(V_S - V_O) * t_{on} = Vo * (T - t_{on})$$
$$\text{Where: } T \equiv t_{on}/\text{Duty\_Cycle}$$

It therefore follows that for MOSFET Q1:

Equation 4-2

$$\text{Duty\_Cycle}_{Q1} = V_O/Vs$$

When choosing an inductor value, a good starting point is to select a value to produce a maximum peak-to-peak ripple current in the inductor equal to ten percent of the maximum load current.

Equation 4-3

$$V = L * (di/dt)$$
$$L = (V_S - V_O) * (t_{on}/I_o * 0.10)$$

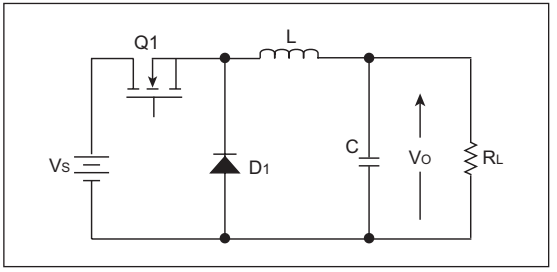
When choosing an output capacitor value, a good starting point is to set the LC filter characteristic impedance equal to the load resistance. This produces an acceptable voltage overshoot when operating at full load and having the load abruptly removed.

Equation 4-4

$$Z_o \equiv \sqrt{L/C}$$
$$C = L/R^2 = (I_O^2 * L)/V_O^2$$

When choosing a diode for D1, choose a device with a sufficient current rating to handle the inductor current during the discharge part of the pulse cycle (I_L).

Figure 4-1: Buck Regulator



Digital Interfacing

When interfacing two devices that operate at different voltages, it is imperative to know the output and input thresholds of both devices. Once these values are known, a technique can be selected for interfacing the devices based on the other requirements of your application. Table 4-1 contains the output and input thresholds that will be used throughout this document. When designing an interface, make sure to reference your manufacturers data sheet for the actual threshold levels.

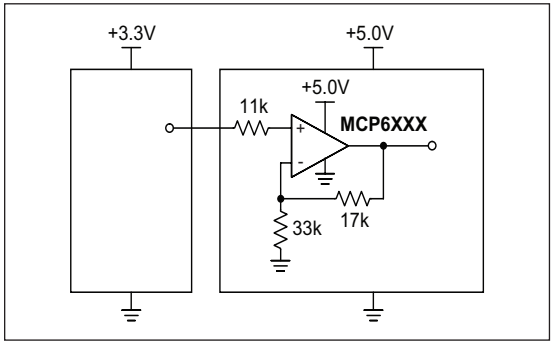
Table 4-1: Input/Output Thresholds

	V_{OH} min	V_{OL} max	V_{IH} min	V_{IL} max
5V TTL	2.4V	0.5V	2.0V	0.8V
3.3V LVTTL	2.4V	0.4V	2.0V	0.8V
5V CMOS	4.7V (Vcc-0.3V)	0.5V	3.5V (0.7xVcc)	1.5V (0.3xVcc)
3.3V LVCMOS	3.0V (Vcc-0.3V)	0.5V	2.3V (0.7xVcc)	1.0V (0.3xVcc)

TIP #14 3.3V → 5V Analog Gain Block

To scale analog voltage up when going from 3.3V supply to 5V supply. The 33 kΩ and 17 kΩ set the op amp gain so that the full scale range is used in both sides. The 11 kΩ resistor limits current back to the 3.3V circuitry.

Figure 14-1: Analog Gain Block



TIP #15 3.3V → 5V Analog Offset Block

Offsetting an analog voltage for translation between 3.3V and 5V.

Shift an analog voltage from 3.3V supply to 5V supply. The 147 kΩ and 30.1 kΩ resistors on the top right and the +5V supply voltage are equivalent to a 0.85V voltage source in series with a 25 kΩ resistor. This equivalent 25 kΩ resistance, the three 25 kΩ resistors, and the op amp form a difference amplifier with a gain of 1 V/V. The 0.85V equivalent voltage source shifts any signal seen at the input up by the same amount; signals centered at 3.3V/2 = 1.65V will also be centered at 5.0V/2 = 2.50V. The top left resistor limits current from the 5V circuitry.

Figure 15-1: Analog Offset Block