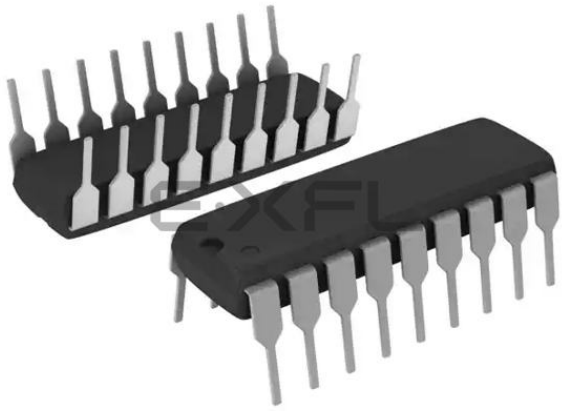


Welcome to E-XFL.COM

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	-
Peripherals	POR, WDT
Number of I/O	13
Program Memory Size	896B (512 x 14)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	80 x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	-
Oscillator Type	External
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Through Hole
Package / Case	18-DIP (0.300", 7.62mm)
Supplier Device Package	18-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16c554-20-p

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.

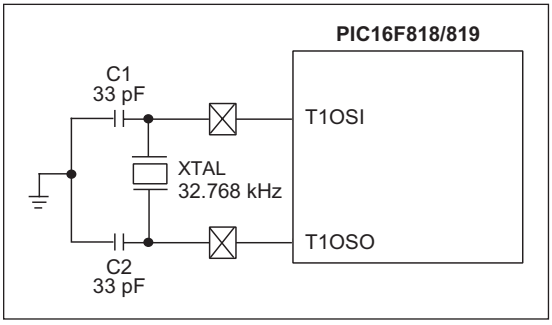
TIP #11 Use Internal RC Oscillators

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 #12 Internal Oscillator Calibration

An internal RC oscillator calibrated from the factory may require further calibration as the temperature or V_{DD} change. Timer1/SOSC can be used to calibrate the internal oscillator by connecting a 32.768 kHz clock crystal. Refer to AN244, “*Internal RC Oscillator Calibration*” for the complete application details. Calibrating the internal oscillator can help save power by allowing for use of the internal RC oscillator in applications which normally require higher accuracy crystals

Figure 12-1: Timer1 Used to Calibrate an Internal Oscillator



The calibration is based on the measured frequency of the internal RC oscillator. For example, if the frequency selected is 4 MHz, we know that the instruction time is 1 µs (F_{OSC}/4) and Timer1 has a period of 30.5 µs (1/32.768 kHz). This means within one Timer1 period, the core can execute 30.5 instructions. If the Timer1 registers are preloaded with a known value, we can calculate the number of instructions that will be executed upon a Timer1 overflow.

This calculated number is then compared against the number of instructions executed by the core. With the result, we can determine if re-calibration is necessary, and if the frequency must be increased or decreased. Tuning uses the OSCTUNE register, which has a ±12% tuning range in 0.8% steps.

TIP #13 Idle and Doze Modes

nanoWatt and nanoWatt XLP devices have an Idle mode where the clock to the CPU is disconnected and only the peripherals are clocked. In PIC16 and PIC18 devices, Idle mode can be entered by setting the Idle bit in the OSCON register to ‘1’ and executing the `SLEEP` instruction. In PIC24, dsPIC® DSCs, and PIC32 devices, Idle mode can be entered by executing the instruction “`PWRSAV #1`”. Idle mode is best used whenever the CPU needs to wait for an event from a peripheral that cannot operate in Sleep mode. Idle mode can reduce power consumption by as much as 96% in many devices.

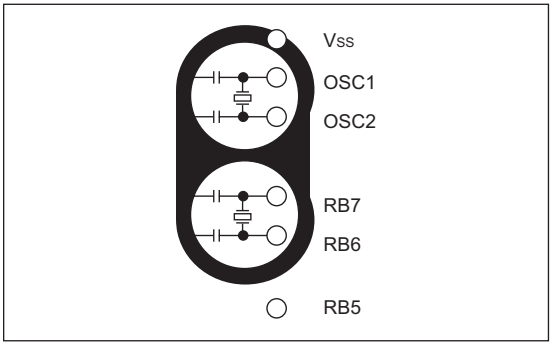
Doze mode is another low power mode available in PIC24, dsPIC DSCs, and PIC32 devices. In Doze mode, the system clock to the CPU is postscaled so that the CPU runs at a lower speed than the peripherals. If the CPU is not tasked heavily and peripherals need to run at high speed, then Doze mode can be used to scale down the CPU clock to a slower frequency. The CPU clock can be scaled down from 1:1 to 1:128. Doze mode is best used in similar situations to Idle mode, when peripheral operation is critical, but the CPU only requires minimal functionality.

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.

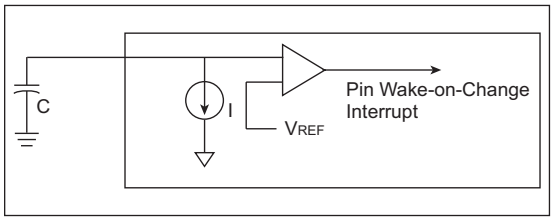
TIP #21 Use Peripheral FIFO and DMA

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 #22 Ultra Low-Power Wake-Up Peripheral

Newer devices have a modification to PORTA that creates an Ultra Low-Power Wake-Up (ULPWU) peripheral. A small current sink and a comparator have been added that allows an external capacitor to be used as a wake-up timer. This feature provides a low-power periodic wake-up source which is dependent on the discharge time of the external RC circuit.

Figure 22-1: Ultra Low-Power Wake-Up Peripheral



If the accuracy of the Watchdog Timer is not required, this peripheral can save a lot of current.

Visit the low power design center at: www.microchip.com/lowpower for additional design resources.

TIP #9 Generating the Time Tick for a RTOS

Real Time Operating Systems (RTOS) require a periodic interrupt to operate. This periodic interrupt, or "tick rate", is the basis for the scheduling system that RTOS's employ. For instance, if a 2 ms tick is used, the RTOS will schedule its tasks to be executed at multiples of the 2 ms. A RTOS also assigns a priority to each task, ensuring that the most critical tasks are executed first. Table 9-1 shows an example list of tasks, the priority of each task and the time interval that the tasks need to be executed.

Table 9-1: Tasks

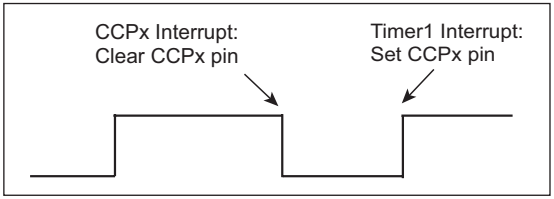
Task	Interval	Priority
Read ADC Input 1	20 ms	2
Read ADC Input 2	60 ms	1
Update LCD	24 ms	2
Update LED Array	36 ms	3
Read Switch	10 ms	1
Dump Data to Serial Port	240 ms	1

The techniques described in Tip #7 can be used to generate the 2 ms periodic interrupt using the CCP module configured in Compare mode.

Note: For more information on RTOSs and their use, see Application Note AN777 "*Multitasking on the PIC16F877 with the Salvo™ RTOS*".

TIP #10 16-Bit Resolution PWM

Figure 10-1: 16-Bit Resolution PWM



1. Configure CCPx to clear output (CCPx pin) on match in Compare mode (CCPxCON <CCPSM3:CCPxM0>).
2. Enable the Timer1 interrupt.
3. Set the period of the waveform via Timer1 prescaler (T1CON <5:4>).
4. Set the duty cycle of the waveform using CCPRxL and CCPRxH.
5. Set CCPx pin when servicing the Timer1 overflow interrupt(1).

Note 1: One hundred percent duty cycle is not achievable with this implementation due to the interrupt latency in servicing Timer1. The period is not affected because the interrupt latency will be the same from period to period as long as the Timer1 interrupt is serviced first in the ISR.

Timer1 has four configurable prescaler values. These are 1:1, 1:2, 1:4 and 1:8. The frequency possibilities of the PWM described above are determined by Equation 10-1.

Equation 10-1

$$F_{PWM} = F_{OSC}/(65536*4*prescaler)$$

For a microcontroller running on a 20 MHz oscillator (Fosc) this equates to frequencies of 76.3 Hz, 38.1 Hz, 19.1 Hz and 9.5 Hz for increasing prescaler values.

COMBINATION CAPTURE AND COMPARE TIPS

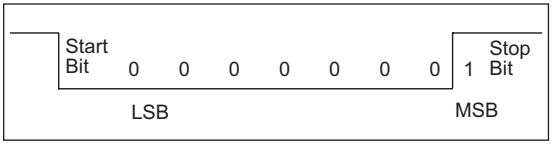
The CCP and ECCP modules can be configured on the fly. Therefore, these modules can perform different functions in the same application provided these functions operate exclusively (not at the same time). This section will provide examples of using a CCP module in different modes in the same application.

TIP #20 RS-232 Auto-baud

RS-232 serial communication has a variety of baud rates to choose from. Multiple transmission rates require software which detects the transmission rate and adjusts the receive and transmit routines accordingly. Auto-baud is used in applications where multiple transmission rates can occur. The CCP module can be configured in Capture mode to detect the baud rate and then be configured in Compare mode to generate or receive RS-232 transmissions.

In order for auto-baud to work, a known calibration character must be transmitted initially from one device to another. One possible calibration character is show in Figure 20-1. Timing this known character provides the device with the baud rate for all subsequent communications.

Figure 20-1: RS-232 Calibration Character



Auto-baud Routine Implementation:

1. Configure CCP module to capture the falling edge (beginning of Start bit).
2. When the falling edge is detected, store the CCPR1 value.
3. Configure the CCP module to capture the rising edge.
4. Once the rising edge is detected, store the CCPR1 value.
5. Subtract the value stored in step 2 from the value in step 4. This is the time for 8 bits.
6. Shift the value calculated in step 5 right 3 times to divide by 8. This result is the period of a bit (T_B).
7. Shift value calculated in step 6 right by 1. This result is half the period of a bit.

The following code segments show the process for transmitting and receiving data in the normal program flow. This same functionality can be accomplished using the CCP module by configuring the module in Compare mode and generating a CCP interrupt every bit period. When this method is used, one bit is either sent or received when the CCP interrupt occurs.

Note: Refer to Application Note AN712 "*RS-232 Auto-baud for the PIC16C5X Devices*" for more details on auto-baud.

NOTES:

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

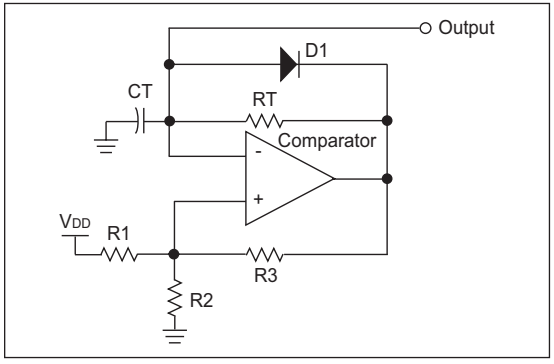
```
Test
   BTFSS CMCON,COUT ;test Cout
   RETLW 0          ;if clear, return 0
   MOVLW CINVBIT    ;if set, invert Cout
   XORWF CMCON,f    ;forces Cout to 0
   BTFSS CMCON,CINV ;test Cout polarity
   RETLW CHNGFLAG   ;if clear, return
                    ;CHNGFLAG
   RETLW COUTMASK + CHNGFLAG;if set,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

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

TIP #9 Multi-Vibrator (Ramp Wave Output)

A multi-vibrator (ramp wave output) is an oscillator designed around a voltage comparator or operational amplifier that produces an asymmetrical output waveform (see Figure 9-1). Resistors R1 through R3 form a hysteresis feedback path from the output to the non-inverting input. Resistor RT, diode D1 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 D1. Because the dynamic on resistance of the diode is significantly lower than RT, the discharge of CT is small when compared to the charge time, and the resulting waveform across CT is a pseudo ramp function with a ramping charge phase and a short-sharp discharge phase.

Figure 9-1: Ramp Waveform Multi-Vibrator



To design this multi-vibrator, first design the hysteresis feedback path using the procedure in Tip #3. Remember that the peak-to-peak amplitude of the ramp wave will be determined by the hysteresis limits. Also, be careful to choose threshold voltages (V_{TH} and V_{TL}) that are evenly spaced within the common mode range of the comparator.

Then use V_{TH} and V_{TL} to calculate values for RT and CT that will result in the desired oscillation frequency F_{OSC}. Equation 9-1 defines the relationship between RT, CT, V_{TH}, V_{TL} and F_{OSC}.

Equation 9-1

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

This assumes that the dynamic on resistance of D1 is much less than RT.

Example:

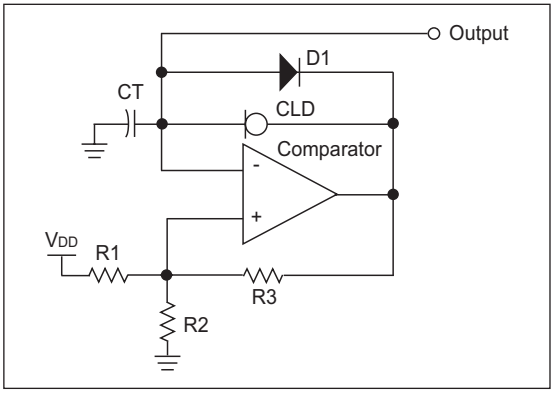
- V_{DD} = 5V, V_{TH} = 1.666V and V_{TH} = 3.333V
- R1, R2 and R3 = 10k
- R_T = 15k, C_T = .1 µF for a F_{OSC} = 906 Hz

Note: Replacing R_T with a current limiting diode will significantly improve the linearity of the ramp wave form. Using the example shown above, a CCL1000 (1 mA Central Semiconductor CLD), will produce a very linear 6 kHz output (see Equation 9-2).

Equation 9-2

$$F_{OSC} = \frac{I_{CLD}}{C\,(V_{TH} - V_{TL})}$$

Figure 9-2: Alternate Ramp Waveform Multi-Vibrator Using a CLD

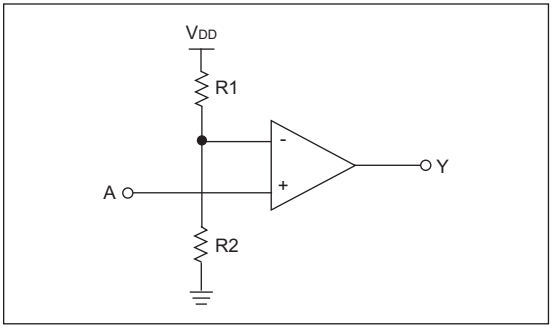


TIP #15 Level Shifter

This tip shows the use of the comparator as a digital logic level shifter. The inverting input is biased to the center of the input voltage range ($V_{IN}/2$). The non-inverting input is then used for the circuit input. When the input is below the $V_{IN}/2$ threshold, the output is low. When the input is above $V_{IN}/2$, then the output is high. Values for R1 and R2 are not critical, though their ratio should result in a threshold voltage $V_{IN}/2$ at the mid-point of the input signal voltage range. Some microcontrollers have the option to connect the inverting input to an internal voltage reference. To use the reference in place of R1 and R2, simply select the internal reference and configure it for one half the input voltage range.

Note: Typical propagation delay for the circuit is 250-350 ns using the typical on-chip comparator peripheral of a microcontroller.

Figure 15-1: Level Shifter



Example:

- V_{IN} = 0 - 2V, $V_{IN}/2$ = 1V, V_{DD} = 5V
- R2 = 10k, R3 = 3.9k

TIP #16 Logic: Inverter

When designing embedded control applications, there is often the need for an external gate. Using the comparator, several simple gates can be implemented. This tip shows the use of the comparator as an inverter.

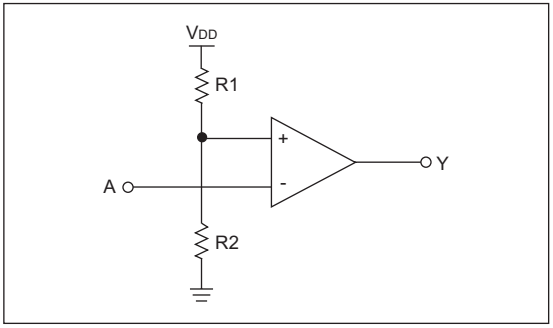
The non-inverting input is biased to the center of the input voltage range, typically $V_{DD}/2$. The inverting input is then used for the circuit input. When the input is below $V_{DD}/2$, the output is high. When the input is above $V_{DD}/2$, then the output is low.

Values for R1 and R2 are not critical, though they must be equal to set the threshold at $V_{DD}/2$.

Some microcontrollers have the option to connect the inverting input to an internal voltage reference. To use the reference in place of R1 and R2, move the input to the non-inverting input and set the output polarity bit in the comparator control register to invert the comparator output.

Note: Typical propagation delay for the circuit is 250-350 ns using the typical on-chip comparator peripheral of a microcontroller.

Figure 16-1: Inverter



TIP #5 Writing a PWM Value to the the CCP Registers With a Mid-Range PIC® Microcontroller

The two PWM LSb’s are located in the CCPCON register of the CCP. This can make changing the PWM period frustrating for a developer. Example 5-1 through Example 5-3 show three macros written for the mid-range product family that can be used to set the PWM period. The first macro takes a 16-bit value and uses the 10 MSb’s to set the PWM period. The second macro takes a 16-bit value and uses the 10 LSb’s to set the PWM period. The last macro takes 8 bits and sets the PWM period. This assumes that the CCP is configured for no more than 8 bits.

Example 5-1: Left Justified 16-Bit Macro

```
pwm_tmp     equ xxx   ;this variable must be
                      ;allocated someplace
setPeriod   macro a   ;a is 2 SFR’s in
                      ;Low:High arrangement
                      ;the 10 MSb’s are the
                      ;desired PWM value
 RRF        a,w       ;This macro will
                      ;change w
 MOVWF      pwm_tmp
 RRF        pwm_tmp,w
 ANDLW      0x30
 IORLW      0x0F
 MOVWF      CCP1CON
 MOVF       a+1,w
 MOVWF      CCPR1L
```

Example 5-2: Right Justified 16-Bit Macro

```
pwm_tmp     equ xxx   ;this variable must be
                      ;allocated someplace
setPeriod   macro a   ;a is 2 bytes in
                      ;Low:High arrangement
                      ;the 10 LSb’s are the
                      ;desired PWM value
 SWAPF      a,w       ;This macro will
                      ;change w
 ANDLW      0x30
 IORLW      0x0F
 MOVWF      CCP1CON
 RLF        a,w
 IORLW      0x0F
 MOVWF      pwm_tmp
 RRF        pwm_tmp,f
 RRF        pwm_tmp,w
 MOVWF      CCPR1L
```

Example 5-3: 8-Bit Macro

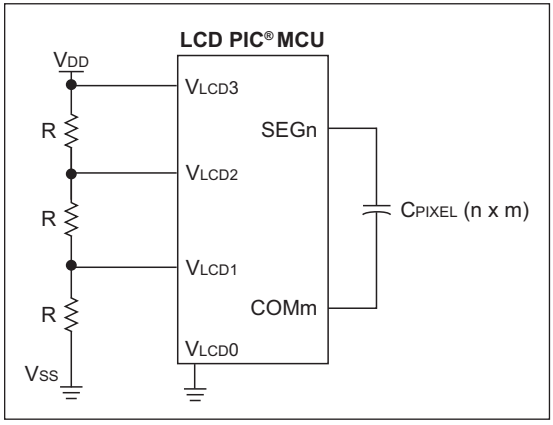
```
pwm_tmp     equ xxx   ;this variable must be
                      ;allocated someplace
setPeriod   macro a   ;a is 1 SFR
 SWAPF      a,w       ;This macro will
                      ;change w
 ANDLW      0x30
 IORLW      0x0F
 MOVWF      CCP1CON
 RRF        a,w
 MOVWF      pwm_tmp
 RRF        pwm_tmp,w
 MOVWF      CCPR1L
```

NOTES:

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

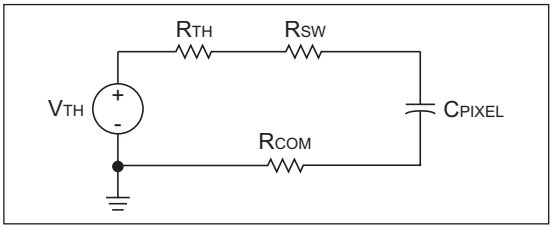


How do we maximize the resistance without adversely effecting the quality of the display? Some basic circuit analysis helps us determine how much we can increase the size of the resistors in the ladder.

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 $V_{LCD}1$.

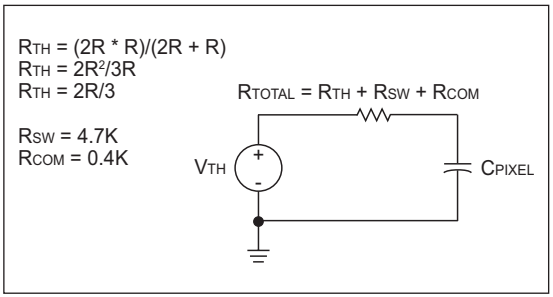
The circuit can be simplified as shown in Figure 3-2. R_{SW} is the resistance of the segment multiplex switch; R_{COM} is the resistance of the common multiplex switch.

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.

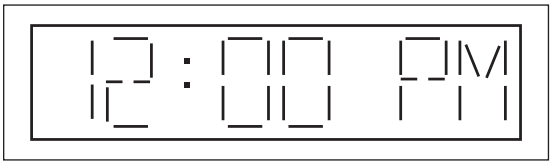
Figure 3-3: LCD Circuit Resistance Estimate



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.

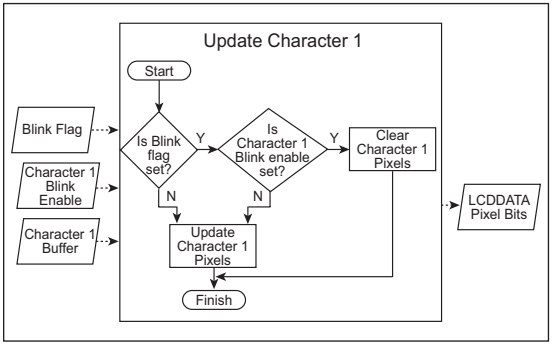
We can see that the time for the voltage across the pixel to change from 0 to V_{TH} will depend on the capacitance of the pixel and the total resistance, of which the resistor ladder Thevenin resistance forms the most significant part.

Figure 11-1: Common Clock Application



Fortunately, blinking is quite easy to implement. There are many ways to implement a blinking effect in software. Any regular event can be used to update a blink period counter. A blink flag can be toggled each time the blink period elapses. Each character or display element that you want to blink can be assigned a corresponding blink enable flag. The flowchart for updating the display would look like:

Figure 11-2: Updating Display Flowchart



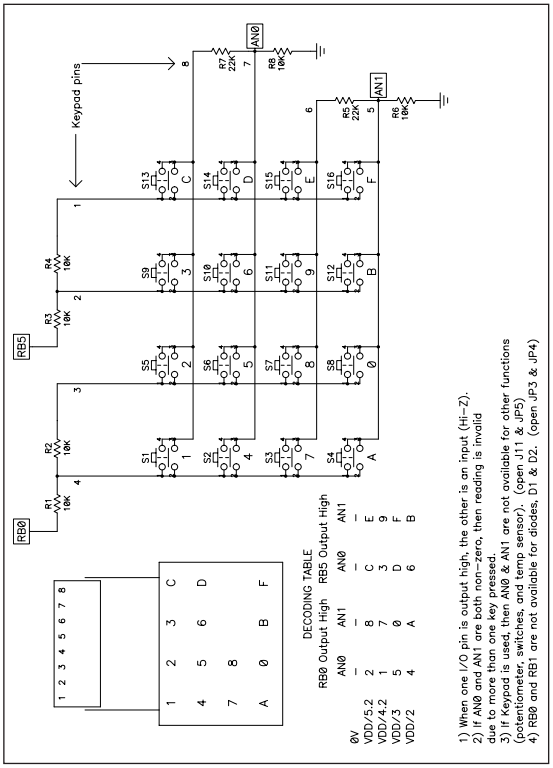
TIP #12 4 x 4 Keypad Interface that Conserves Pins for LCD Segment Drivers

A typical digital interface to a 4 x 4 keypad uses 8 digital I/O pins. But using eight pins as digital I/Os can take away from the number of segment driver pins available to interface to an LCD.

By using 2 digital I/O pins and 2 analog input pins, it is possible to add a 4 x 4 keypad to the PIC microcontroller without sacrificing any of its LCD segment driver pins.

The schematic for keypad hook-up is shown in Figure 12-1. This example uses the PIC18F8490, but the technique could be used on any of the LCD PIC MCUs.

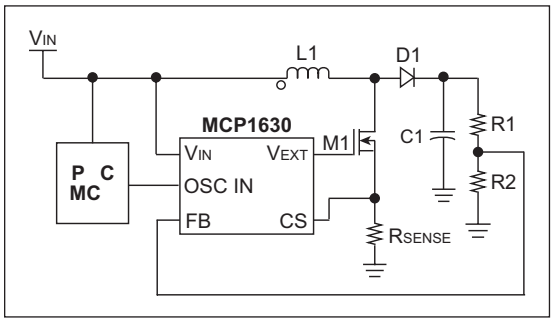
Figure 12-1: Keypad Hook-up Schematic



TIP #5 Using a PIC® Microcontroller as a Clock Source for a SMPS PWM Generator

A PIC MCU can be used as the clock source for a PWM generator, such as the MCP1630.

Figure 5-1: PIC MCU and MCP1630 Example Boost Application



The MCP1630 begins its cycle when its clock/oscillator source transitions from high-to-low, causing its PWM output to go high state. The PWM pulse can be terminated in any of three ways:

1. The sensed current in the magnetic device reaches 1/3 of the error amplifier output.
2. The voltage at the Feedback (FB) pin is higher than the reference voltage (V_{REF}).
3. The clock/oscillator source transitions from low-to-high.

The switching frequency of the MCP1630 can be adjusted by changing the frequency of the clock source. The maximum on-timer of the MCP1630 PWM can be adjusted by changing the duty cycle of the clock source.

The PIC MCU has several options for providing this clock source:

- The Fosc/4 pin can be enabled. This will produce a 50% duty cycle square wave that is 1/4th of the oscillator frequency. Tip #4 provides both example software and information on clock dithering using the $F_{OSC}/4$ output.
- For PIC MCUs equipped with a Capture/Compare/PWM (CCP) or Enhanced CCP (ECCP) module, a variable frequency, variable duty cycle signal can be created with little software overhead. This PWM signal is entirely under software control and allows advanced features, such as soft-start, to be implemented using software.
- For smaller parts that do not have a CCP or ECCP module, a software PWM can be created. Tips #1 and #2 use software PWM for soft-start and provide software examples.

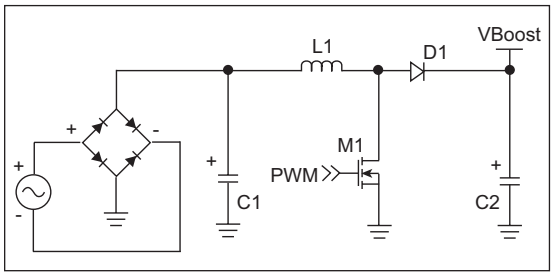
TIP #7 Using a PIC® Microcontroller for Power Factor Correction

In AC power systems, the term Power Factor (PF) is used to describe the fraction of power actually used by a load compared to the total apparent power supplied.

Power Factor Correction (PFC) is used to increase the efficiency of power delivery by maximizing the PF.

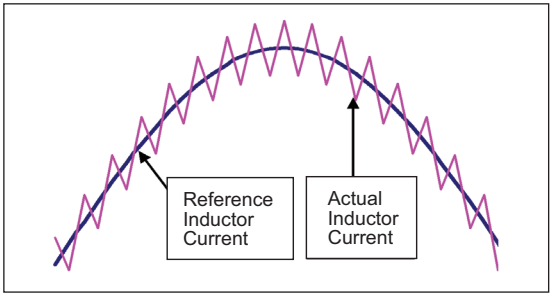
The basis for most Active PFC circuits is a boost circuit, shown in Figure 7-1.

Figure 7-1: Typical Power Factor Correction Boost Supply



The AC voltage is rectified and boosted to voltages as high as 400 VDC. The unique feature of the PFC circuit is that the inductor current is regulated to maintain a certain PF. A sine wave reference current is generated that is in phase with the line voltage. The magnitude of the sine wave is inversely proportional to the voltage at VBoost. Once the sine wave reference is established, the inductor current is regulated to follow it, as shown in Figure 7-2.

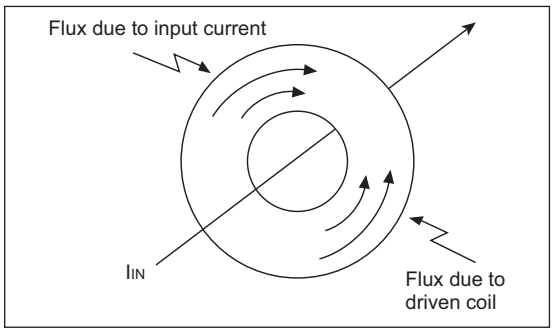
Figure 7-2: Desired and Actual Inductor Currents



A PIC MCU has several features that allow it to perform power factor correction.

- The PIC MCUs CCP module can be used to generate a PWM signal that, once filtered, can be used to generate the sine wave reference signal.
- The PIC Analog-to-Digital (A/D) converter can be used to sense VBoost and the reference sine wave can be adjusted in software.
- The interrupt-on-change feature of the PIC MCU input pins can be used to allow the PIC MCU to synchronize the sine wave reference to the line voltage by detecting the zero crossings.
- The on-chip comparators can be used for driving the boost MOSFET(s) using the PWM sine wave reference as one input and the actual inductor current as another.

Figure 15-2: Flux Directions



The net flux in the core should be approximately zero. Because the flux will always be very near zero, the core will be very linear over the small operating range.

When I_{IN} = 0, the output of the comparator will have an approximate 50% duty cycle. As the current moves one direction, the duty cycle will increase. As the current moves the other direction, the duty cycle will decrease. By measuring the duty cycle of the resulting comparator output, we can determine the value of I_{IN}.

Finally, a Delta-Sigma ADC can be used to perform the actual measurement. Features such as comparator sync and Timer1 gate allow the Delta-Sigma conversion to be taken care of entirely in hardware. By taking 65,536 (2^16) samples and counting the number of samples that the comparator output is low or high, we can obtain a 16-bit A/D result.

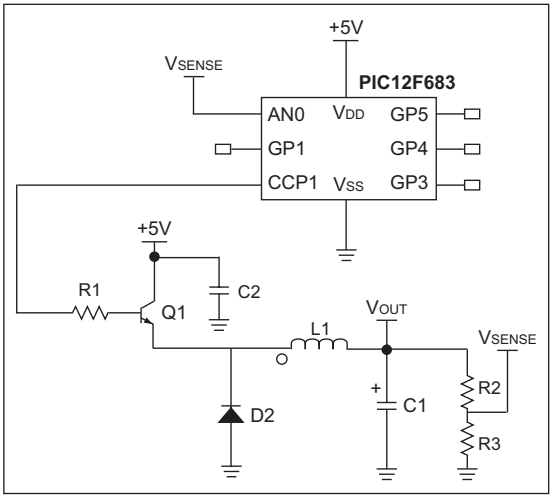
Example schematic and software are provided for the PIC12F683 in both C and Assembly.

For more information on using a PIC MCU to implement a Delta-Sigma converter, please refer to AN700, “*Make a Delta-Sigma Converter Using a Microcontroller’s Analog Comparator Module*” (DS00700), which includes example software.

TIP #16 Implementing a PID Feedback Control in a PIC12F683-Based SMPS Design

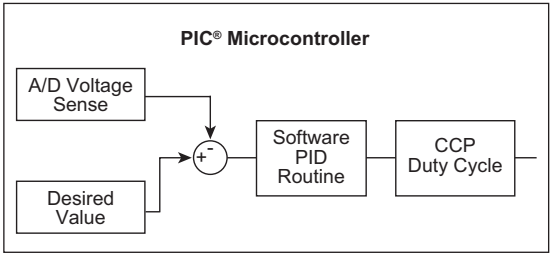
Simple switching power supplies can be controlled digitally using a Proportional Integral Derivative (PID) algorithm in place of an analog error amplifier and sensing the voltage using the Analog-to-Digital Converter (ADC).

Figure 16-1: Simple PID Power Supply



The design in Figure 16-1 utilizes an 8-pin PIC12F683 PIC MCU in a buck topology. The PIC12F683 has the basic building blocks needed to implement this type of power supply: an A/D converter and a CCP module.

Figure 16-2: PID Block Diagram



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

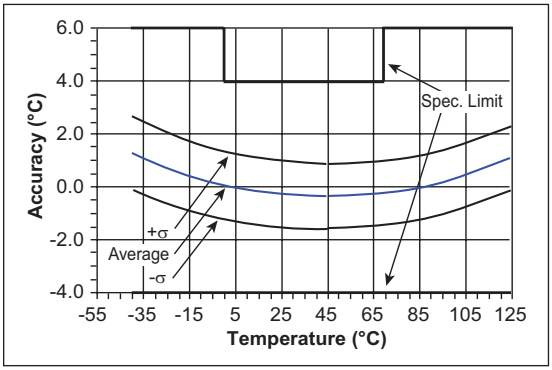


Figure 20-1 shows the accuracy of a 100 sample lot of MCP9700 temperature sensors. Despite the fact that the sensor’s error is nonlinear, a PIC microcontroller (MCU) can be used to compensate the sensor’s reading.

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

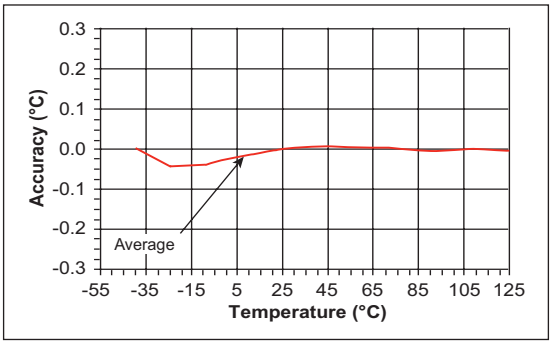


Figure 20-2 shows the average accuracy for the 100 sample lot of MCP9700 temperature sensors after compensation. The average error has been decreased over the full temperature range.

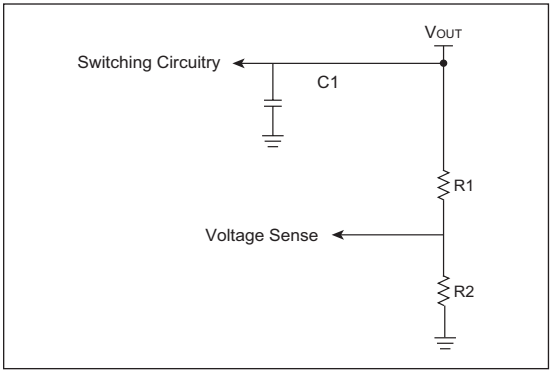
It is also possible to compensate for error from voltage references using this method.

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

TIP #21 Using Output Voltage Monitoring to Create a Self-Calibration Function

A PIC microcontroller can be used to create a switching power supply controlled by a PID loop (as described in Tip #16). This type of power supply senses its output voltage digitally, compares that voltage to the desired reference voltage and makes duty cycle changes accordingly. Without calibration, it is sensitive to component tolerances.

Figure 21-1: Typical Power Supply Output Stage



The output stage of many power supplies is similar to Figure 21-1. R1 and R2 are used to set the ratio of the voltage that is sensed and compared to the reference.

A simple means of calibrating this type of power supply is as follows:

1. Supply a known reference voltage to the output of the supply.
2. Place the supply in Calibration mode and allow it to sense that reference voltage.

By providing the supply with the output voltage that it is to produce, it can then sense the voltage across the resistor divider and store the sensed value. Regardless of resistor tolerances, the sensed value will always correspond to the proper output value for that particular supply.

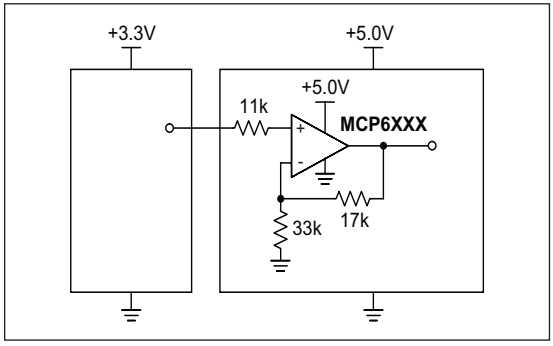
Futhermore, this setup could be combined with Tip #20 to calibrate at several temperatures.

This setup could also be used to create a programmable power supply by changing the supplied reference and the resistor divider for voltage feedback.

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