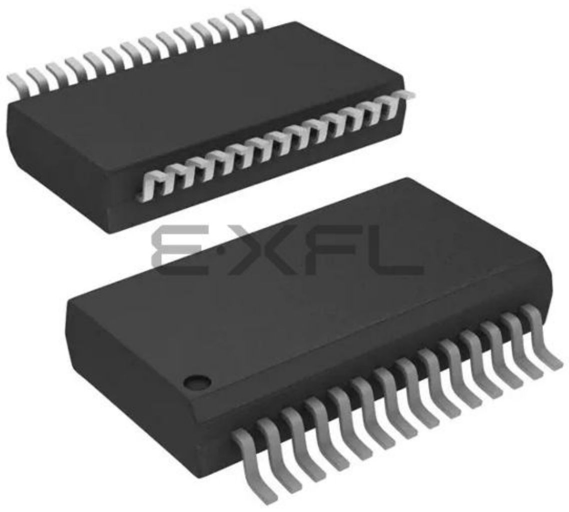
Microchip Technology - PIC16C63A-20/SS Datasheet



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"

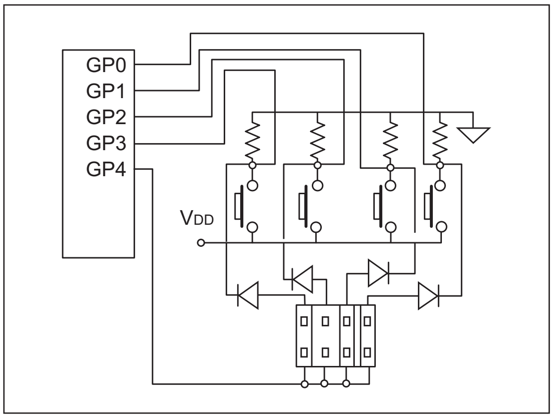
Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	22
Program Memory Size	7KB (4K x 14)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	192 x 8
Voltage - Supply (Vcc/Vdd)	4V ~ 5.5V
Data Converters	-
Oscillator Type	External
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SSOP (0.209", 5.30mm Width)
Supplier Device Package	28-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16c63a-20-ss

Tips ‘n Tricks Table of Contents

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.

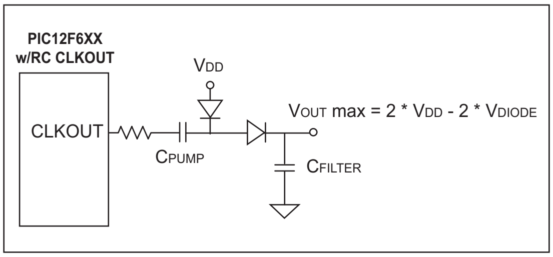
Figure 9-1



- 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 #13 Reading a Sensor With Higher Accuracy

Sensors can be read directly with the A/D but in some applications, factors such as temperature, external component accuracy, sensor non-linearity and/or decreasing battery voltage need to be considered. In other applications, more than 10 bits of accuracy are needed and a slower sensor read is acceptable. The following tips deal with these factors and show how to get the most out of a PIC MCU.

13.1. RC Timing Method (with reference resistor)

13.2. Charge Balancing Method

13.3. A/D Method

Tip #13.1 Reading a Sensor With Higher Accuracy – RC Timing Method

RC Timing Method:

Simple RC step response

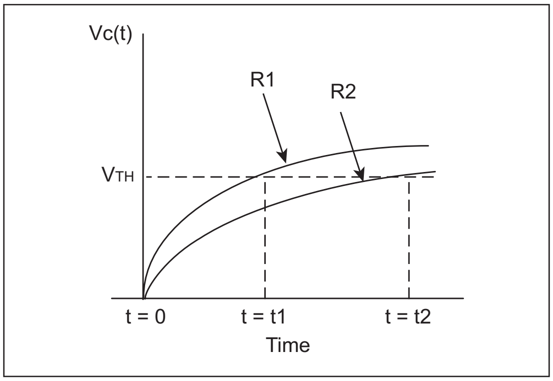
$V_C(t) = V_{DD} * (1 - e^{-t/(RC)})$

$t = -RC \ln(1 - V_{TH}/V_{DD})$

V_{TH}/V_{DD} is constant

$R2 = (t2/t1) * R1$

Figure 13-1



A reference resistor can be used to improve the accuracy of an analog sensor reading. In this diagram, the charge time of a resistor/capacitor combination is measured using a timer and a port input or comparator input switches from a ‘0’ to ‘1’. The R1 curve uses a reference resistor and the R2 curve uses the sensor. The charge time of the R1 curve is known and can be used to calibrate the unknown sensor reading, R2. This reduces the affects of temperature, component tolerance and noise while reading the sensor.

CHAPTER 2
PIC® Microcontroller Low Power 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 an wide range of everyday products, from smoke detectors, hospital ID tags and pet containment systems, to industrial, automotive and medical products.

PIC MCUs featuring nanoWatt technology implement a variety of important features which have become standard in PIC microcontrollers. Since the release of nanoWatt technology, changes in MCU process technology and improvements in performance have resulted in new requirements for lower power. PIC MCUs with nanoWatt eXtreme Low Power (nanoWatt XLP™) improve upon the original nanoWatt technology by dramatically reducing static power consumption and providing new flexibility for dynamic power management.

The following series of Tips n’ Tricks can be applied to many applications to make the most of PIC MCU nanoWatt and nanoWatt XLP devices.

GENERAL LOW POWER TIPS ‘N TRICKS

The following tips can be used with all PIC MCUs to reduce the power consumption of almost any application.

CHAPTER 3
PIC® Microcontroller CCP and ECCP Tips ‘n Tricks

Table Of Contents

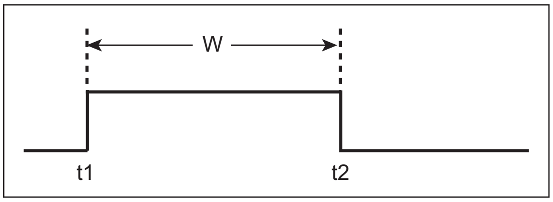
TIPS ‘N TRICKS INTRODUCTION

Microchip continues to provide innovative products that are smaller, faster, easier-to-use and more reliable. PIC® microcontrollers (MCUs) are used in a wide range of everyday products, from washing machines, garage door openers and television remotes to industrial, automotive and medical products.

The Capture, Compare and PWM (CCP) modules that are found on many of Microchip’s microcontrollers are used primarily for the measurement and control of time-based pulse signals. The Enhanced CCP (ECCP), available on some of Microchip’s devices, differs from the regular CCP module in that it provides enhanced PWM functionality – namely, full-bridge and half-bridge support, programmable dead-band delay and enhanced PWM auto-shutdown. The ECCP and CCP modules are capable of performing a wide variety of tasks. This document will describe some of the basic guidelines to follow when using these modules in each mode, as well as give suggestions for practical applications.

TIP #3 Measuring Pulse Width

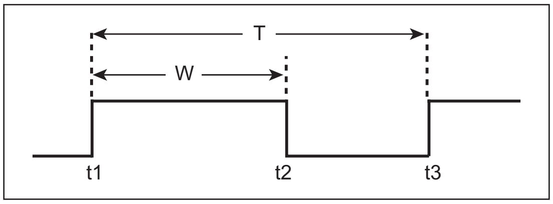
Figure 3-1: Pulse Width



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 #7 Periodic Interrupts

Generating interrupts at periodic intervals is a useful technique implemented in many applications. This technique allows the main loop code to run continuously, and then, at periodic intervals, jump to the interrupt service routine to execute specific tasks (i.e., read the ADC). Normally, a timer overflow interrupt is adequate for generating the periodic interrupt. However, sometimes it is necessary to interrupt at intervals that can not be achieved with a timer overflow interrupt. The CCP configured in Compare mode makes this possible by shortening the full 16-bit time period.

Example Problem:

A PIC16F684 running on its 8 MHz internal oscillator needs to be configured so that it updates a LCD exactly 5 times every second.

Step #1: Determine a Timer1 prescaler that allows an overflow at greater than 0.2 seconds

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

b) For a prescaler of 1:1, Timer1 overflows in 32.8 ms.

c) A prescaler of 8 will cause an overflow at a time greater than 0.2 seconds.
8 x 32.8 ms = 0.25s

Step #2: Calculate CCPR1 (CCPR1L and CCPR1H) to shorten the time-out to exactly 0.2 seconds

a) CCPR1 = Interval Time/(TOSC*4*prescaler) = 0.2/(125 ns*4*8) = 5000 = 0xC350

b) Therefore, CCPR1L = 0x50, and CCPR1H = 0xC3

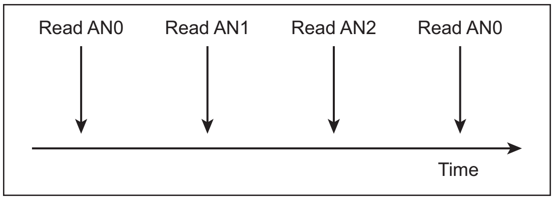
Step #3: Configuring CCP1CON

The CCP module should be configured in Trigger Special Event mode. This mode generates an interrupt when the Timer1 equals the value specified in CCPR1L and Timer1 is automatically cleared(1). For this mode, CCP1CON = ‘`b00001011`’.

Note 1: Trigger Special Event mode also starts an A/D conversion if the A/D module is enabled. If this functionality is not desired, the CCP module should be configured in “generate software interrupt-on-match only” mode (i.e., CCP1CON = b‘`00001010`’). Timer 1 must also be cleared manually during the CCP interrupt.

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

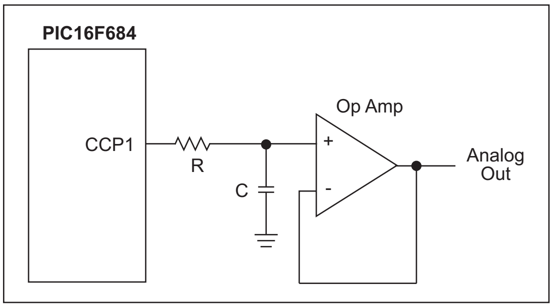
The ECCP module should be configured in Trigger Special Event mode. This mode generates an interrupt when Timer1 equals the value specified in CCPR1. Timer1 is automatically cleared and the GO bit in ADCON0 is automatically set. For this mode, CCP1CON = ‘`b00001011`’.

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.

TIP #16 Generating an Analog Output

Figure 16-1: Low-Pass Filter



Pulse-width modulated signals can be used to create Digital-to-Analog (D/A) converters with only a few external components. Conversion of PWM waveforms to analog signals involves the use of an analog low-pass filter. In order to eliminate unwanted harmonics caused by a PWM signal to the greatest degree possible, the frequency of the PWM signal (F_{PWM}) should be significantly higher than the bandwidth (F_{BW}) of the desired analog signal. Equation 16-1 shows this relation.

Equation 16-1

$$F_{PWM} = K{*}F_{BW}$$

Where harmonics decrease as K increases

R and C are chosen based on the following equation:

Equation 16-2

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

Pick a value of C arbitrarily and then calculate R. The attenuation of the PWM frequency for a given RC filter is:

Equation 16-3

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

If the attenuation calculated in Equation 16-3 is not sufficient, then K must be increased in Equation 16-1. See Application Note AN538 “*Using PWM to Generate Analog Output in PIC17C42*” for more details on using PWM to generate an analog output.

CHAPTER 5
DC Motor Control Tips ‘n Tricks

Table Of Contents

TIPS ‘N TRICKS INTRODUCTION

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

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

TIP #4 Drive Software

Pulse-Width Modulation (PWM) Algorithms

Pulse-Width Modulation is critical to modern digital motor controls. By adjusting the pulse width, the speed of a motor can be efficiently controlled without larger linear power stages. Some PIC devices and all dsPIC DSCs have hardware PWM modules on them. These modules are built into the Capture/Compare/PWM (CCP) peripheral. CCP peripherals are intended for a single PWM output, while the Enhanced CCP (ECCP) is designed to produce the complete H-Bridge output for bidirectional Brushed DC motor control. If cost is a critical design point, a PIC device with a CCP module may not be available, so software generated PWM is a good alternative.

The following algorithms are designed to efficiently produce an 8-bit PWM output on the Mid-Range family of PIC microcontrollers. These algorithms are implemented as macros. If you want these macros to be a subroutine in your program, simply remove the macro statements and replace them with a label and a return statement.

Example 4-1: 1 Output 8-Bit PWM

```
pwm_counter equ xxx ;variable
pwm         equ xxx ;variable

set_pwm macro A     ;sets the pwm
                    ;setpoint to the
                    ;value A
 MOVLW A
 MOVWF pwm
 endm

update_PWM macro    ;performs one update
                    ;of the PWM signal
                    ;place the PWM output
                    ;pin at bit 0 or 7 of
                    ;the port
 MOVF pwm_counter,w
 SUBWF pwm, w       ;if the output
                    ;is on bit 0
 RLF        PORTC,f ;replace PORTC with
                    ;the correct port if
                    ;the output is on bit
                    ;7 of the port
                    ;replace the rlf with
                    ;rrf incf
                    ;pwm_counter,f
```

Example 4-2: 8 Output 8-Bit PWM

```
pwm_counter equ xxx      ;variable
pwm0        equ xxx      ;
pwm1        equ pwm0+1
pwm2        equ pwm1+1
pwm3        equ pwm2+1
pwm4        equ pwm3+1
pwm5        equ pwm4+1
pwm6        equ pwm5+1
pwm7        equ pwm6+1
output      equ pwm7+1
set_pwm macro A,b        ;sets pwm b with
                         ;the value A
 MOVLW pwm0
 ADDLW b
 MOVWF fsr
 MOVLW a
 MOVWF indf
 endm

update_PWM macro         ;peforms one
                         ;update of all 8
                         ;PWM signals
                         ;all PWM signals
                         ;must be on the
                         ;same port
 MOVF       pwm_counter,w
 SUBWF      pwm0,w
 RLF        output,f
 MOVF       pwm_counter,w
 SUBWF      pwm1,w
 RLF        output,f
 MOVF       pwm_counter,w
 SUBWF      pwm2,w
 RLF        output,f
 MOVF       pwm_counter,w
 SUBWF      pwm3,w
 RLF        output,f
 MOVF       pwm_counter,w
 SUBWF      pwm4,w
 RLF        output,f
 MOVF       pwm_counter,w
 SUBWF      pwm5,w
 RLF        output,f
 MOVF       pwm_counter,w
 SUBWF      pwm6,w
 RLF        output,f
 MOVF       pwm_counter,w
 SUBWF      pwm7,w
 RLF        output,w
 MOVWF      PORTC
 INCF       pwm_counter,f
endm
```

NOTES:

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.

Application Note References

- AN220, "*Watt-Hour Meter Using PIC16C923 and CS5460*" (DS00220)
- AN582, "*Low-Power Real-Time Clock*" (DS00582)
- AN587, "*Interfacing PIC® MCUs to an LCD Module*" (DS00587)
- AN649, "*Yet Another Clock Featuring the PIC16C924*" (DS00649)
- AN658, "*LCD Fundamentals Using PIC16C92X Microcontrollers*" (DS00658)
- TB084, "*Contrast Control Circuits for the PIC16F91X*" (DS91084)

Application notes can be found on the Microchip web site at www.microchip.com.

CHAPTER 7
Intelligent Power Supply Design 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. PIC® microcontrollers (MCUs) are used in a wide range of everyday products from washing machines, garage door openers and television remotes to industrial, automotive and medical products.

While some designs such as Switch Mode Power Supplies (SMPS) are traditionally implemented using a purely analog control scheme, these designs can benefit from the configurability and intelligence that can only be realized by adding a microcontroller.

This document showcases several examples in which a PIC microcontroller may be used to increase the functionality of a design with a minimal increase in cost.

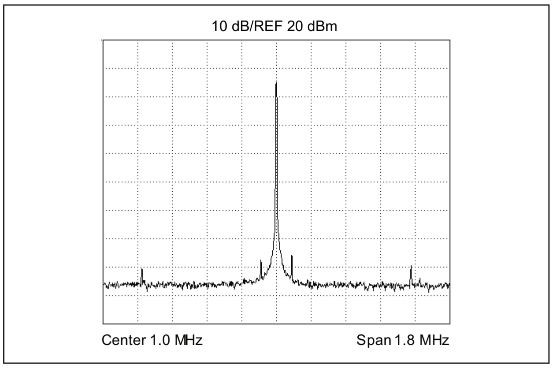
Several of the tips provide working software examples or reference other documents for more information. The software and referenced documents can be found on the Microchip web site at www.microchip.com/tipsntricks.

TIP #4 Creating a Dithered PWM Clock

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

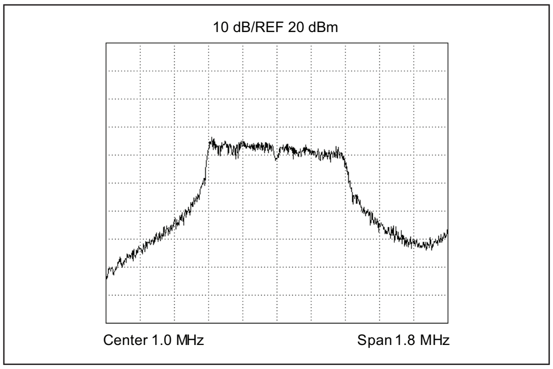
The PIC10F200 has an internal 4 MHz oscillator. A scaled version of oscillator can be output on a pin (Fosc/4). The scaled output is 1/4 of the oscillator frequency (1 MHz) and will always have a 50% duty cycle. Figure 4-1 shows a spectrum analyzer shot of the output of the Fosc/4 output.

Figure 4-1: Spectrum of Clock Output Before Dithering



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

Figure 4-2: Spectrum of Clock Output After Dithering



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

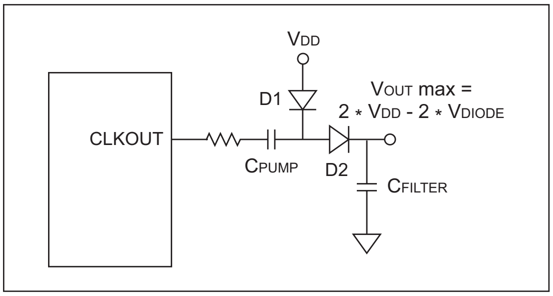
Example software is provided for the PIC10F200 that performs the pseudo-random sequence generation and loads the OSCCAL register.

TIP #10 Driving High Side FETs

In applications where high side N channel FETs are to be driven, there are several means for generating an elevated driving voltage. One very simple method is to use a voltage doubling charge pump as shown in Figure 10-1.

Method 1

Figure 10-1: Typical Change Pump

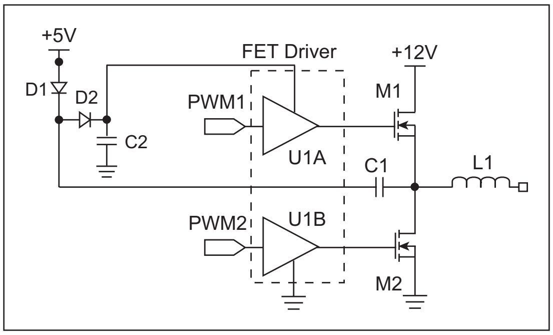


The PIC MCUs CLKOUT pin toggles at 1/4 of the oscillator frequency. When CLKOUT is low, D1 is forward biased and conducts current, thereby charging C_{PUMP}. After CLKOUT is high, D2 is forward biased, moving the charge to C_{FILTER}. The result is a voltage equal to twice the V_{DD} minus two diode drops. This can be used with a PWM or any other I/O pin that toggles.

In Figure 10-2, a standard FET driver is used to drive both the high and low side FETs by using the diode and capacitor arrangement.

Method 2

Figure 10-2: Schematic



The +5V is used for powering the microcontroller. Using this arrangement, the FET driver would have approximately 12 + (5 - V_{DIODE}) - V_{DIODE} volts as a supply and is able to drive both the high and low side FETs.

The circuit above works by charging C1 through D1 to (5V - V_{DIODE}) while M2 is on, effectively connecting C1 to ground. When M2 turns off and M1 turns on, one side of C1 is now at 12V and the other side is at 12V + (5V - V_{DIODE}). The D2 turns on and the voltage supplied to the FET driver is 12V + (5V - V_{DIODE}) - V_{DIODE}.

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

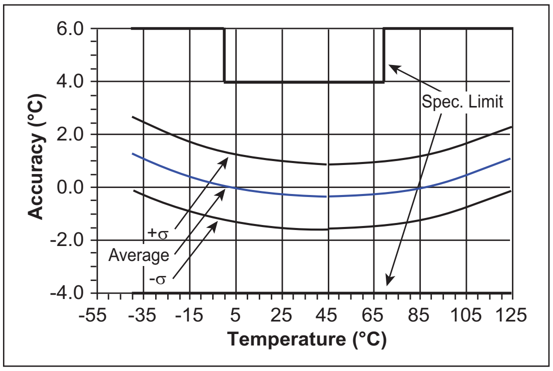


Figure 20-1 shows the accuracy of a 100 sample lot of MCP9700 temperature sensors. Despite the fact that the sensor’s error is nonlinear, a PIC microcontroller (MCU) can be used to compensate the sensor’s reading.

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

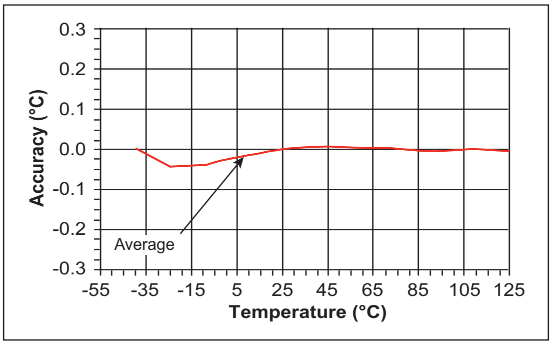


Figure 20-2 shows the average accuracy for the 100 sample lot of MCP9700 temperature sensors after compensation. The average error has been decreased over the full temperature range.

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).

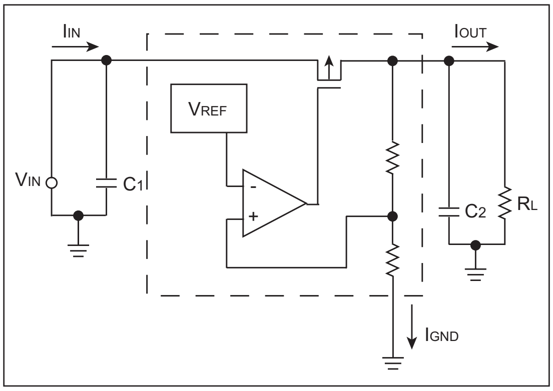
TIP #1 Powering 3.3V Systems From 5V Using an LDO Regulator

The dropout voltage of standard three-terminal linear regulators is typically 2.0-3.0V. This precludes them from being used to convert 5V to 3.3V reliably. Low Dropout (LDO) regulators, with a dropout voltage in the few hundred milli-volt range, are perfectly suited for this type of application. Figure 1-1 contains a block diagram of a basic LDO system with appropriate current elements labeled. From this figure it can be seen that an LDO consists of four main elements:

1. Pass transistor
2. Bandgap reference
3. Operational amplifier
4. Feedback resistor divider

When selecting an LDO, it is important to know what distinguishes one LDO from another. Device quiescent current, package size and type are important device parameters. Evaluating for each parameter for the specific application yields an optimal design.

Figure 1-1: LDO Voltage Regulator



An LDOs quiescent current, I_Q, is the device ground current, I_{GND}, while the device is operating at no load. I_{GND} is the current used by the LDO to perform the regulating operation. The efficiency of an LDO can be approximated as the output voltage divided by the input voltage when $I_{OUT} >> I_Q$. However, at light loads, the I_Q must be taken into account when calculating the efficiency. An LDO with lower I_Q will have a higher light load efficiency. This increase in light load efficiency has a negative effect on the LDO performance. Higher quiescent current LDOs are able to respond quicker to sudden line and load transitions.

TIP #16 5V → 3.3V Active Analog Attenuator

Reducing a signal’s amplitude from a 5V to 3.3V system using an op amp.

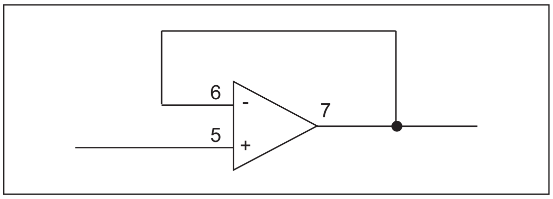
The simplest method of converting a 5V analog signal to a 3.3V analog signal is to use a resistor divider with a ratio R1:R2 of 1.7:3.3. However, there are a few problems with this.

1. The attenuator may be feeding a capacitive load, creating an unintentional low pass filter.
2. The attenuator circuit may need to drive a low-impedance load from a high-impedance source.

Under either of these conditions, an op amp becomes necessary to buffer the signals.

The op amp circuit necessary is a unity gain follower (see Figure 16-1).

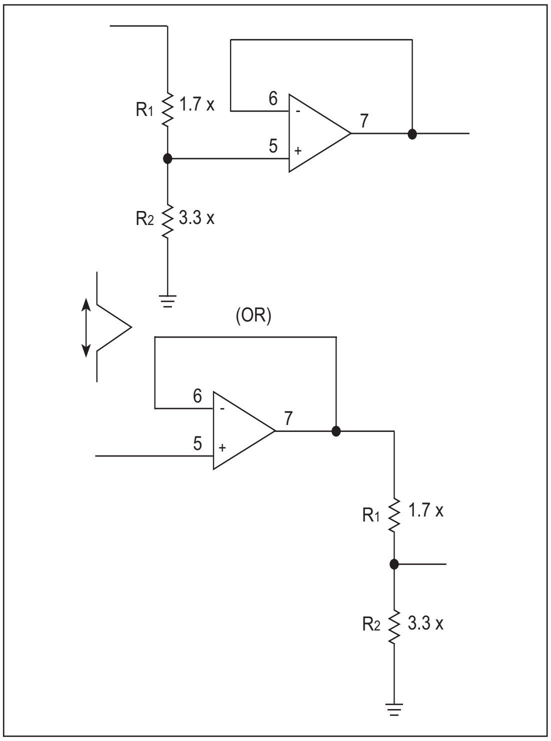
Figure 16-1: Unity Gain



This circuit will output the same voltage that is applied to the input.

To convert the 5V signal down to a 3V signal, we simply add the resistor attenuator.

Figure 16-2: Op Amp Attenuators



If the resistor divider is before the unity gain follower, then the lowest possible impedance is provided for the 3.3V circuits. Also, the op amp can be powered from 3.3V, saving some power. If the X is made very large, then power consumed by the 5V side can be minimized.

If the attenuator is added after the unity gain follower, then the highest possible impedance is presented to the 5V source. The op amp must be powered from 5V and the impedance at the 3V side will depend upon the value of R1||R2.