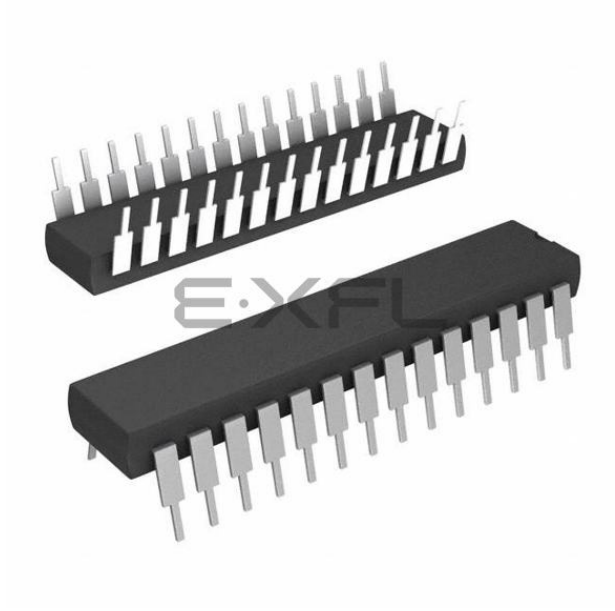


Welcome to E-XFL.COM

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	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.300", 7.62mm)
Supplier Device Package	28-SPDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16c63a-20i-sp

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Tips ‘n Tricks Table of Contents

CHAPTER 1
8-Pin Flash PIC® Microcontrollers Tips ‘n Tricks

Table Of Contents

TIPS ‘N TRICKS INTRODUCTION

Microchip continues to provide innovative products that are smaller, faster, easier to use and more reliable. The 8-pin Flash PIC® microcontrollers (MCU) are used in an wide range of everyday products, from toothbrushes, hair dryers and rice cookers to industrial, automotive and medical products.

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

TIPS ‘N TRICKS WITH HARDWARE

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

TIP #3 Read Three States From One Pin

To check state Z:

- Drive output pin high
- Set to Input
- Read 1
- Drive output pin low
- Set to Input
- Read 0

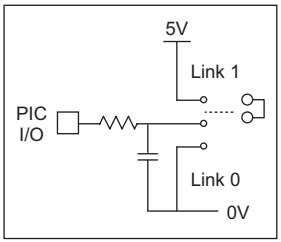
To check state 0:

- Read 0 on pin

To check state 1:

- Read 1 on pin

Figure 3-1



State	Link 0	Link 1
0	closed	open
1	open	closed
NC	open	open

Jumper has three possible states: not connected, Link 1 and Link 0. The capacitor will charge and discharge depending on the I/O output voltage allowing the "not connected" state. Software should check the "not connected" state first by driving I/O high, reading 1 and driving I/O low and reading 0. The "Link 1" and "Link 0" states are read directly.

TIP #4 Reading DIP Switches

The input of a timer can be used to test which switch(s) is closed. The input of Timer1 is held high with a pull-up resistor. Sequentially, each switch I/O is set to input and Timer1 is checked for an increment indicating the switch is closed.

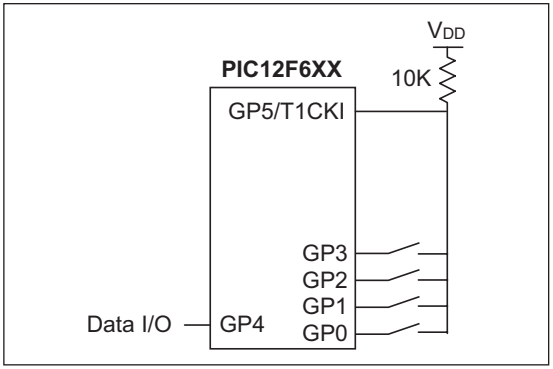
Example 4-1

```
        movlw   b'11111111'
        movwf   TRISIO
        movwf   DIP
        movlw   b'00000111'
        movwf   T1CON
        movlw   b'11111110'
        movwf   Mask
        clrf    GPIO
LOOP
        clrf    TMR1L
        movf    Mask,W
        movwf   TRISIO
        btfsc   TMR1L,0
        andwf   DIP,F
        bsf     STATUS,C
        rlf     Mask,F
        btfsc   Mask,4
        goto    Loop
        retlw   0
```

Each bit in the DP register represents its corresponding switch position. By setting Timer1 to FFFFh and enabling its interrupt, an increment will cause a rollover and generate an interrupt. This will simplify the software by eliminating the bit test on the TMR1L register.

Sequentially set each GPIO to an input and test for TMR1 increment (or 0 if standard I/O pin is used).

Figure 4-1



TIPS ‘N TRICKS WITH SOFTWARE

To reduce costs, designers need to make the most of the available program memory in MCUs. Program memory is typically a large portion of the MCU cost. Optimizing the code helps to avoid buying more memory than needed. Here are some ideas that can help reduce code size.

TIP #15 Delay Techniques

- Use GOTO “next instruction” instead of two NOPs.
- Use CALL Rtrn as quad, 1 instruction NOP (where “Rtrn” is the exit label from existing subroutine).

Example 15-1

```
        NOP
        NOP             ;2 instructions, 2 cycles

        GOTO    $+1     ;1 instruction, 2 cycles

        CALL Rtrn       ;1 instruction, 4 cycles
        . . .
Rtrn    RETURN
```

MCUs are commonly used to interface with the “outside world” by means of a data bus, LEDs, buttons, latches, etc. Because the MCU runs at a fixed frequency, it will often need delay routines to meet setup/hold times of other devices, pause for a handshake or decrease the data rate for a shared bus.

Longer delays are well-suited for the DECFSZ and INCFSZ instructions where a variable is decremented or incremented until it reaches zero when a conditional jump is executed. For shorter delays of a few cycles, here a few ideas to decrease code size.

For a two-cycle delay, it is common to use two NOP instructions which uses two program memory locations. The same result can be achieved by using “goto $+1”. The “$” represents the current program counter value in MPASM™ Assembler. When this instruction is encountered, the MCU will jump to the next memory location. This is what it would have done if two NOP’s were used but since the GOTO instruction uses two instruction cycles to execute, a two-cycle delay was created. This created a two-cycle delay using only one location of program memory.

To create a four-cycle delay, add a label to an existing RETURN instruction in the code. In this example, the label “Rtrn” was added to the RETURN of subroutine that already existed somewhere in the code. When executing “CALL Rtrn”, the MCU delays two instruction cycles to execute the CALL and two more to execute the RETURN. Instead of using four NOP instructions to create a four-cycle delay, the same result was achieved by adding a single CALL instruction.

TIP #3 Configuring Port Pins

All PIC MCUs have bidirectional I/O pins. Some of these pins have analog input capabilities. It is very important to pay attention to the signals applied to these pins so the least amount of power will be consumed.

Unused Port Pins

If a port pin is unused, it may be left unconnected but configured as an output pin driving to either state (high or low), or it may be configured as an input with an external resistor (about 10 kΩ) pulling it to V_{DD} or V_{SS}. If configured as an input, only the pin input leakage current will be drawn through the pin (the same current would flow if the pin was connected directly to V_{DD} or V_{SS}). Both options allow the pin to be used later for either input or output without significant hardware modifications.

Digital Inputs

A digital input pin consumes the least amount of power when the input voltage is near V_{DD} or V_{SS}. If the input voltage is near the midpoint between V_{DD} and V_{SS}, the transistors inside the digital input buffer are biased in a linear region and they will consume a significant amount of current. If such a pin can be configured as an analog input, the digital buffer is turned off, reducing both the pin current as well as the total controller current.

Analog Inputs

Analog inputs have a very high-impedance so they consume very little current. They will consume less current than a digital input if the applied voltage would normally be centered between V_{DD} and V_{SS}. Sometimes it is appropriate and possible to configure digital inputs as analog inputs when the digital input must go to a low power state.

Digital Outputs

There is no additional current consumed by a digital output pin other than the current going through the pin to power the external circuit. Pay close attention to the external circuits to minimize their current consumption.

TIP #4 Use High-Value Pull-Up Resistors

It is more power efficient to use larger pull-up resistors on I/O pins such as MCLR, I^2C™ signals, switches and for resistor dividers. For example, a typical I^2C pull-up is 4.7k. However, when the I^2C is transmitting and pulling a line low, this consumes nearly 700 uA of current for each bus at 3.3V. By increasing the size of the I^2C pull-ups to 10k, this current can be halved. The tradeoff is a lower maximum I^2C bus speed, but this can be a worthwhile trade in for many low power applications. This technique is especially useful in cases where the pull-up can be increased to a very high resistance such as 100k or 1M.

TIP #5 Reduce Operating Voltage

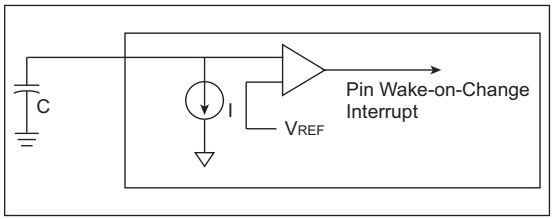
Reducing the operating voltage of the device, V_{DD}, is a useful step to reduce the overall power consumption. When running, power consumption is mainly influenced by the clock speed. When sleeping, the most significant factor is leakage in the transistors. At lower voltages, less charge is required to switch the system clocks and transistors leak less current.

It is important to pay attention to how reducing the operating voltage reduces the maximum allowed operating frequency. Select the optimum voltage that allows the application to run at its maximum speed. Refer to the device data sheet for the maximum operating frequency of the device at the given voltage.

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.

ECCP/CCP Register Listing

	Capture Mode	Compare Mode	PWM Mode
CCPxCON	Select mode	Select mode	Select mode, LSB of duty cycle
CCPRxL	Timer1 capture (LSB)	Timer1 compare (LSB)	MSB of duty cycle
CCPRxH	Timer1 capture (MSB)	Timer1 compare (MSB)	N/A
TRISx	Set CCPx pin to input	Set CCPx pin to output	Set CCPx pin(s) to output(s)
T1CON	Timer1 on, prescaler	Timer1 on, prescaler	N/A
T2CON	N/A	N/A	Timer2 on, prescaler
PR2	N/A	N/A	Timer2 period
PIE1	Timer1 interrupt enable	Timer1 interrupt enable	Timer2 interrupt enable
PIR1	Timer1 interrupt flag	Timer1 interrupt flag	Timer2 interrupt flag
INTCON	Global/ peripheral interrupt enable	Global/ peripheral interrupt enable	Global/ peripheral interrupt enable
PWM1CON(1)	N/A	N/A	Set dead band, auto-restart control
ECCPAS(1)	N/A	N/A	Auto-shutdown control

Note 1: Only on ECCP module.

CAPTURE TIPS ‘N TRICKS

In Capture mode, the 16-bit value of Timer1 is captured in CCPRxH:CCPRxL when an event occurs on pin CCPx. An event is defined as one of the following and is configured by CCPxCON<3:0>:

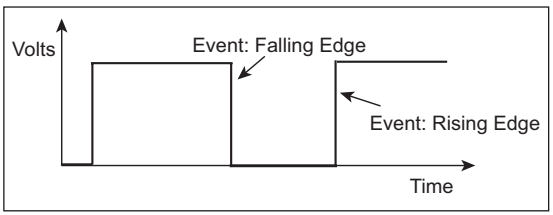
- Every falling edge
- Every rising edge
- Every 4th rising edge
- Every 16th rising edge

“When Would I Use Capture Mode?”

Capture mode is used to measure the length of time elapsed between two events. An event, in general, is either the rising or falling edge of a signal (see Figure 1 “Defining Events”).

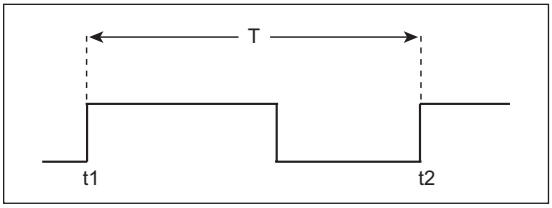
An example of an application where Capture mode is useful is reading an accelerometer. Accelerometers typically vary the duty cycle of a square wave in proportion to the acceleration acting on a system. By configuring the CCP module in Capture mode, the PIC microcontroller can measure the duty cycle of the accelerometer with little intervention on the part of the microcontroller firmware. Tip #4 goes into more detail about measuring duty cycle by configuring the CCP module in Capture mode.

Figure 1: Defining Events



TIP #1 Measuring the Period of a Square Wave

Figure 1-1: Period



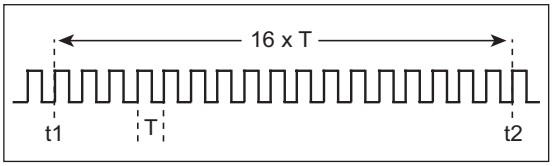
1. Configure control bits CCPxM3:CCPxM0 (CCPxCON<3:0>) to capture every rising edge of the waveform.
2. Configure the Timer1 prescaler so Timer1 with run T_{MAX}(1) without overflowing.
3. Enable the CCP interrupt (CCPxIE bit).
4. When a CCP interrupt occurs:
   a) Subtract saved captured time (t1) from captured time (t2) and store (use Timer1 interrupt flag as overflow indicator).
   b) Save captured time (t2).
   c) Clear Timer1 flag if set.

The result obtained in step 4.a is the period (T).

Note 1: T_{MAX} is the maximum pulse period that will occur.

TIP #2 Measuring the Period of a Square Wave with Averaging

Figure 2-1: Period Measurement



1. Configure control bits CCPxM3:CCPxM0 (CCPxCON<3:0>) to capture every 16th rising edge of the waveform.
2. Configure the Timer1 prescaler so Timer1 will run 16 T_{MAX}(1) without overflowing.
3. Enable the CCP interrupt (CCPxIE bit).
4. When a CCP interrupt occurs:
   a) Subtract saved captured time (t1) from captured time (t2) and store (use Timer1 interrupt flag as overflow indicator).
   b) Save captured time (t2).
   c) Clear Timer1 flag if set.
   d) Shift value obtained in step 4.a right four times to divide by 16 – this result is the period (T).

Note 1: T_{MAX} is the maximum pulse period that will occur.

The following are the advantages of this method as opposed to measuring the periods individually.

- Fewer CCP interrupts to disrupt program flow
- Averaging provides excellent noise immunity

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 #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.

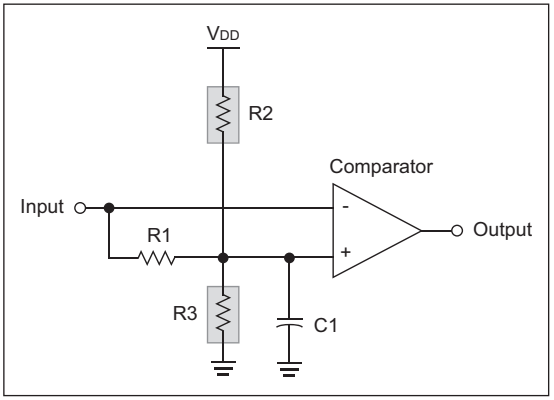
TIP #6 Data Slicer

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

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

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

Figure 6-1: Data Slicer



Example:

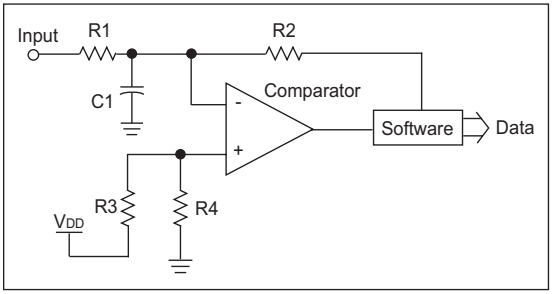
Data rate of 10 kbits/second. A low pass filter frequency of 500 Hz: R1 = 10k, C1 = 33 μF. R2 or R3 should be 500k to 1 MB.

TIP #14 Delta-Sigma ADC

This tip describes the creation of a hardware/software-based Delta-Sigma ADC. A Delta-Sigma ADC is based on a Delta-Sigma modulator composed of an integrator, a comparator, a clock sampler and a 1-bit DAC output. In this example, the integrator is formed by R1 and C1. The comparator is an on-chip voltage comparator. The clock sampler is implemented in software and the 1-bit DAC output is a single I/O pin. The DAC output feeds back into the integrator through R2.

Resistors R3 and R4 form a $V_{DD}/2$ reference for the circuit (see Figure 14-1).

Figure 14-1: Delta-Sigma Modulator



In operation, the feedback output from the software is a time sampled copy of the comparator output. In normal operation, the modulator output generates a PWM signal which is inversely proportional to the input voltage. As the input voltage increases, the PWM signal will drop in duty cycle to compensate. As the input decreases, the duty cycle rises.

To perform an A-to-D conversion, the duty cycle must be integrated over time, digitally, to integrate the duty cycle to a binary value. The software starts two counters. The first counts the total number of samples in the conversion and the second counts the number of samples that were low. The ratio of the two counts is equal to the ratio of the input voltage over V_{DD}.

> **Note:** This assumes that R1 and R2 are equal and R3 is equal to R4. If R1 and R2 are not equal, then the input voltage is also scaled by the ratio of R2 over R1, and R3 must still be equal to R4.

For a more complete description of the operation of a Delta-Sigma ADC and example firmware, see Application Note AN700 “*Make A Delta-Sigma Converter Using a Microcontroller’s Analog Comparator Module*.”

Example:

- R3 = R4 = 10 kHz
- R1 = R2 = 5.1k
- C1 = 1000 pF

TIP #2 Brushless DC Motor Drive Circuits

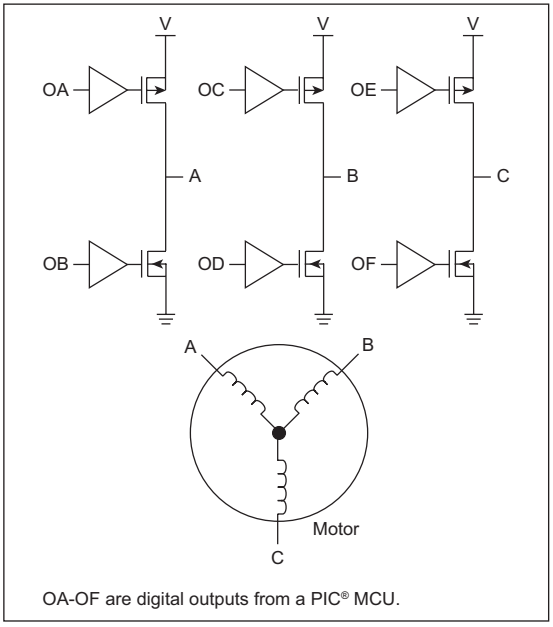
A Brushless DC motor is a good example of simplified hardware increasing the control complexity. The motor cannot commutate the windings (switch the current flow), so the control circuit and software must control the current flow correctly to keep the motor turning smoothly. The circuit is a simple half-bridge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

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

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

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

Figure 2-1: 3 Phase Brushless DC Motor Control



OA-OF are digital outputs from a PIC® MCU.

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

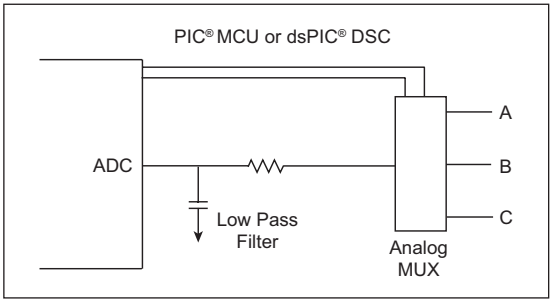


Figure 3-1: 4 and 5 Wire Stepper Motors

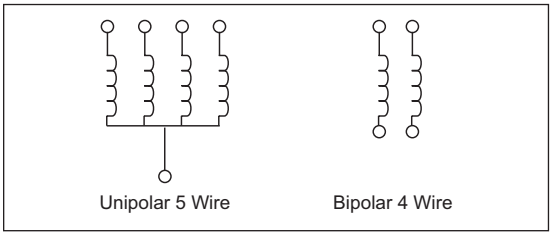


Figure 3-2: 6 and 8 Wire Stepper Motors

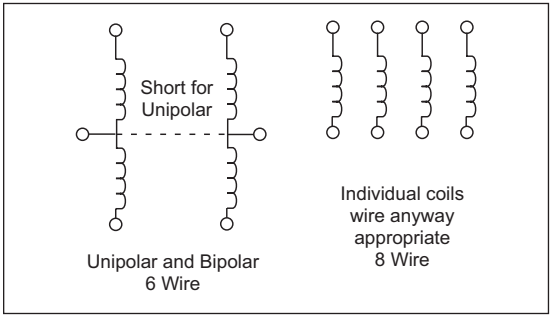


Figure 3-3: Unipolar Motor (4 Low Side Switches)

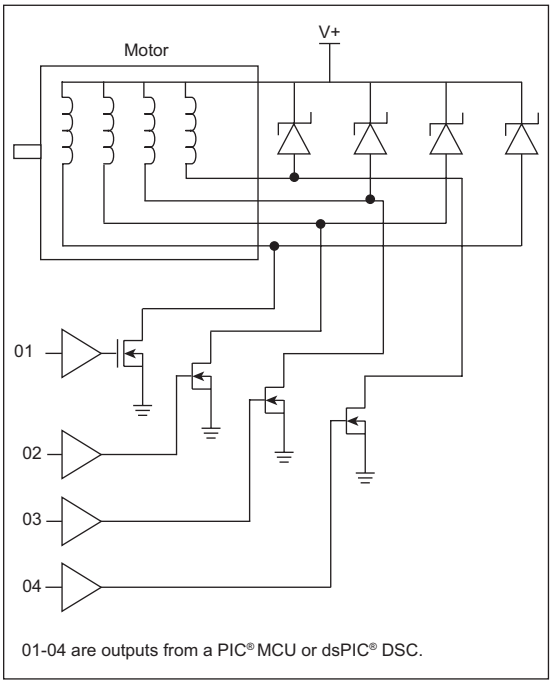


Figure 3-4: Bipolar Motor (4 Half-Bridges)

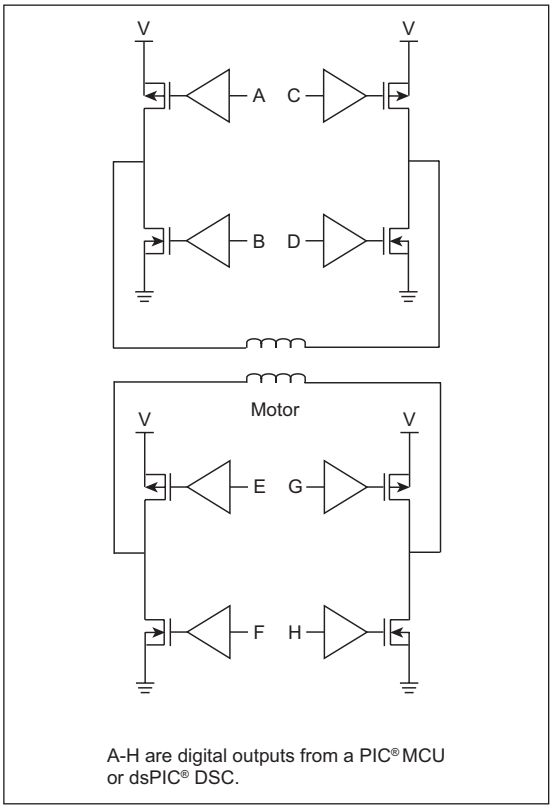
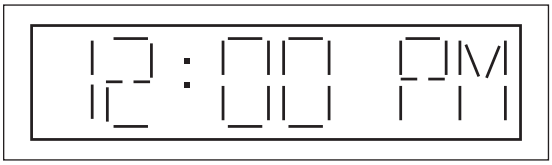
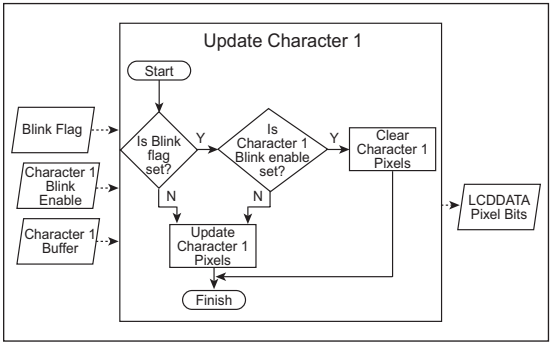


Figure 11-1: Common Clock Application



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



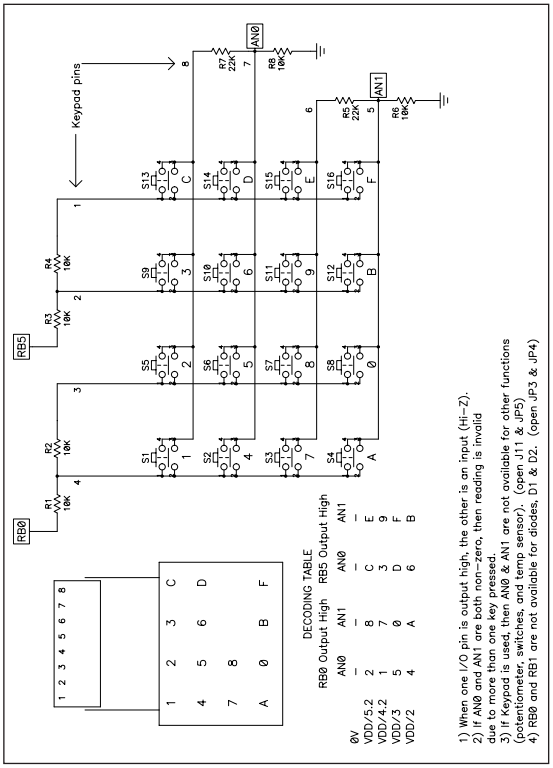
TIP #12 4 x 4 Keypad Interface that Conserves Pins for LCD Segment Drivers

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 #7 3.3V → 5V Using a Diode Offset

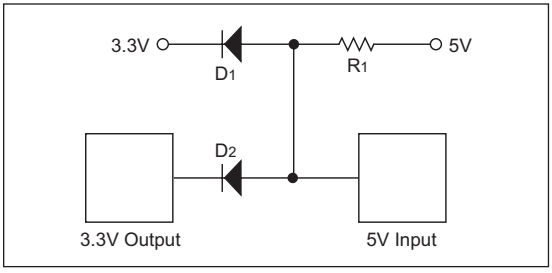
The inputs voltage thresholds for 5V CMOS and the output drive voltage for 3.3V LVTTL and LVCMOS are listed in Table 7-1.

Table 7-1: Input/Output Thresholds

	5V CMOS Input	3.3V LVTTL Output	3.3V LVCMOS Output
High Threshold	> 3.5V	> 2.4V	> 3.0V
Low Threshold	< 1.5V	< 0.4V	< 0.5V

Note that both the high and low threshold input voltages for the 5V CMOS inputs are about a volt higher than the 3.3V outputs. So, even if the output from the 3.3V system could be offset, there would be little or no margin for noise or component tolerance. What is needed is a circuit that offsets the outputs and increases the difference between the high and low output voltages.

Figure 7-1: Diode Offset



When output voltage specifications are determined, it is done assuming that the output is driving a load between the output and ground for the high output, and a load between 3.3V and the output for the low output. If the load for the high threshold is actually between the output and 3.3V, then the output voltage is actually much higher as the load resistor is the mechanism that is pulling the output up, instead of the output transistor.

If we create a diode offset circuit (see Figure 7-1), the output low voltage is increased by the forward voltage of the diode D1, typically 0.7V, creating a low voltage at the 5V CMOS input of 1.1V to 1.2V. This is well within the low threshold input voltage for the 5V CMOS input. The output high voltage is set by the pull-up resistor and diode D2, tied to the 3.3V supply. This puts the output high voltage at approximately 0.7V above the 3.3V supply, or 4.0 to 4.1V, which is well above the 3.5V threshold for the 5V CMOS input.

Note: For the circuit to work properly, the pull-up resistor must be significantly smaller than the input resistance of the 5V CMOS input, to prevent a reduction in the output voltage due to a resistor divider effect at the input. The pull-up resistor must also be large enough to keep the output current loading on the 3.3V output within the specification of the device.

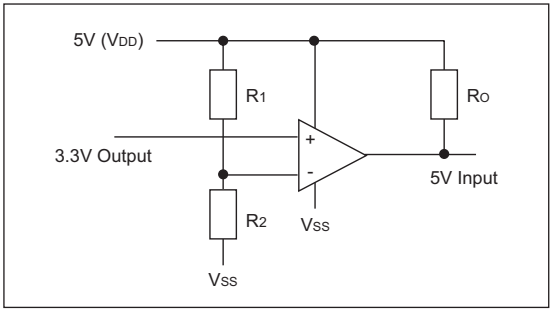
TIP #8 3.3V → 5V Using a Voltage Comparator

The basic operation of the comparator is as follows:

- When the voltage at the inverting (-) input is greater than that at the non-inverting (+) input, the output of the comparator swings to V_{SS}.
- When the voltage at the non-inverting (+) input is greater than that at the non-inverting (-) input, the output of the comparator is in a high state.

To preserve the polarity of the 3.3V output, the 3.3V output must be connected to the non-inverting input of the comparator. The inverting input of the comparator is connected to a reference voltage determined by R1 and R2, as shown in Figure 8-1.

Figure 8-1: Comparator Translator



Calculating R1 and R2

The ratio of R1 and R2 depends on the logic levels of the input signal. The inverting input should be set to a voltage halfway between V_{OL} and V_{OH} for the 3.3V output. For an LVCMOS output, this voltage is:

Equation 8-1:

$$1.75V = \frac{(3.0V + .5V)}{2}$$

Given that R1 and R2 are related by the logic levels:

Equation 8-2:

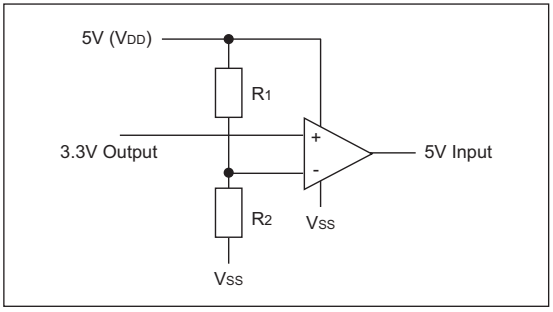
$$R_1 = R_2 \left(\frac{5V}{1.75V} - 1 \right)$$

assuming a value of 1K for R2, R1 is 1.8K.

An op amp wired up as a comparator can be used to convert a 3.3V input signal to a 5V output signal. This is done using the property of the comparator that forces the output to swing high (V_{DD}) or low (V_{SS}), depending on the magnitude of difference in voltage between its 'inverting' input and 'non-inverting' input.

Note: For the op amp to work properly when powered by 5V, the output must be capable of rail-to-rail drive.

Figure 8-2: Op Amp as a Comparator



TIP #16 5V → 3.3V Active Analog Attenuator

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

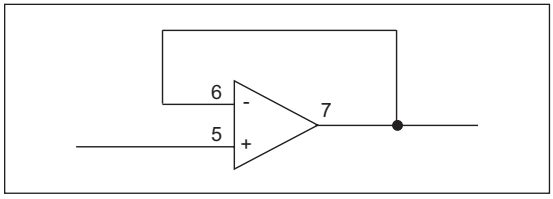
The simplest method of converting a 5V analog signal to a 3.3V analog signal is to use a resistor divider with a ratio R1:R2 of 1.7:3.3. However, there are a few problems with this.

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).

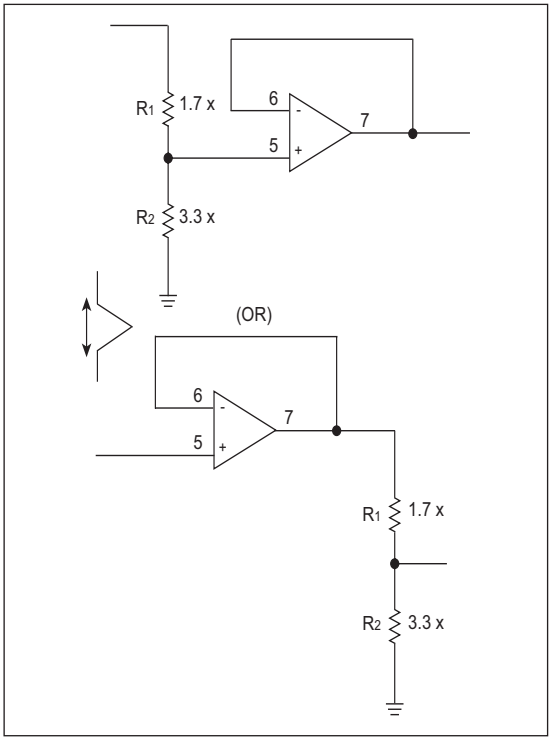
Figure 16-1: Unity Gain



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



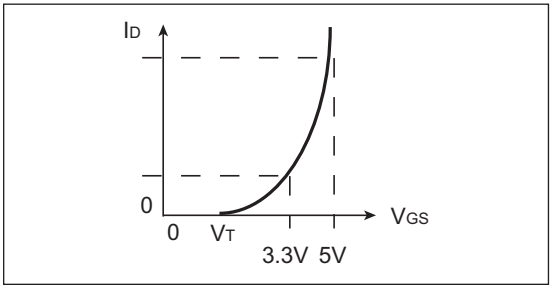
If the resistor divider is before the unity gain follower, then the lowest possible impedance is provided for the 3.3V circuits. Also, the op amp can be powered from 3.3V, saving some power. If the X is made very large, then power consumed by the 5V side can be minimized.

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.

TIP #19 Driving N-Channel MOSFET Transistors

Care must be taken when selecting an external N-Channel MOSFET for use with a 3.3V microcontroller. The MOSFET gate threshold voltage is an indication of the device's capability to completely saturate. For 3.3V applications, select MOSFETs that have an ON resistance rating for gate drive of 3V or less. For example, a FET that is rated for 250 µA of drain current with 1V applied from gate-to-source is not necessarily going to deliver satisfactory results for 100 mA load with a 3.3V drive. When switching from 5V to 3V technology, review the gate-to-source threshold and ON resistance characteristics very carefully as shown in Figure 19-1. A small decrease in gate drive voltage can significantly reduce drain current.

Figure 19-1: Drain Current Capability Versus Gate to Source Voltage



Low threshold devices commonly exist for MOSFETs with drain-to-source voltages rated below 30V. MOSFETs with drain-to-source voltages above 30V typically have higher gate thresholds (VT).

Table 19-1: RDS(ON) and VGS(th) Specifications for IRF7467

RDS(on)	Static Drain-to-Source On-Resistance	–	9.4	12	mΩ	VGS = 10V, ID = 11A
		–	10.6	13.5		VGS = 4.5V, ID = 9.0A
		–	17	35		VGS = 2.8V, ID = 5.5A
VGS(th)	Gate Threshold Voltage	0.6	–	2.0	V	VDS = VGS, ID = 250 µA

As shown in Table 19-1, the threshold voltage for this 30V, N-Channel MOSFET switch is 0.6V. The resistance rating for this MOSFET is 35 mΩ with 2.8V applied gate, as a result, this device is well suited for 3.3V applications.

Table 19-2: RDS(ON) and VGS(th) Specifications for IRF7201

RDS(on)	Static Drain-to-Source On-Resistance	–	–	0.030	Ω	VGS = 10V, ID = 7.3A
		–	–	0.050		VGS = 4.5V, ID = 3.7A
VGS(th)	Gate Threshold Voltage	1.0	–	–	V	VDS = VGS, ID = 250 µA

For the IRF7201 data sheet specifications, the gate threshold voltage is specified as a 1.0V minimum. This does not mean the device can be used to switch current with a 1.0V gate-to-source voltage as there is no RDS(ON) specification for VGS(th) values below 4.5V. This device is not recommended for 3.3V drive applications that require low switch resistance but can be used for 5V drive applications.