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	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/pic16c63at-20i-ss

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

Table Of Contents

TIPS ‘N TRICKS WITH HARDWARE

TIP #1: Dual Speed RC Oscillator 1-2
TIP #2: Input/Output Multiplexing..... 1-2
TIP #3: Read Three States From One Pin.... 1-3
TIP #4: Reading DIP Switches..... 1-3
TIP #5: Scanning Many Keys With One Input..... 1-4
TIP #6: Scanning Many Keys and Wake-up From Sleep..... 1-4
TIP #7: 8x8 Keyboard with 1 Input..... 1-5
TIP #8: One Pin Power/Data..... 1-5
TIP #9: Decode Keys and ID Settings 1-6
TIP #10: Generating High Voltages 1-6
TIP #11: VDD Self Starting Circuit..... 1-7
TIP #12: Using PIC® MCU A/D For Smart Current Limiter..... 1-7
TIP #13: Reading A Sensor With Higher Accuracy..... 1-8
TIP #13.1: Reading A Sensor With Higher Accuracy – RC Timing Method..... 1-8
TIP #13.2: Reading A Sensor With Higher Accuracy – Charge Balancing Method 1-10
TIP #13.3: Reading A Sensor With Higher Accuracy – A/D Method..... 1-11
TIP #14: Delta Sigma Converter..... 1-11

TIPS ‘N TRICKS WITH SOFTWARE

TIP #15: Delay Techniques..... 1-12
TIP #16: Optimizing Destinations..... 1-13
TIP #17: Conditional Bit Set/Clear 1-13
TIP #18: Swap File Register with W 1-14
TIP #19: Bit Shifting Using Carry Bit..... 1-14

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.

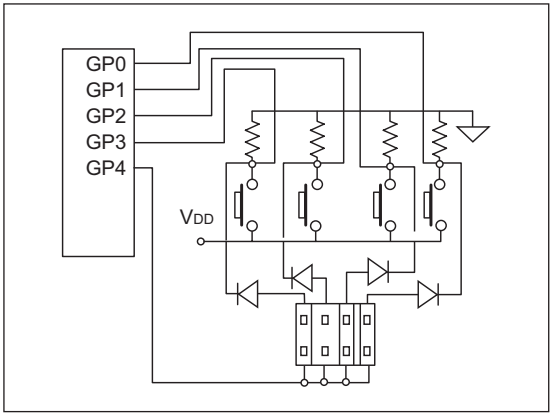
TIPS ‘N TRICKS WITH HARDWARE

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 #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.

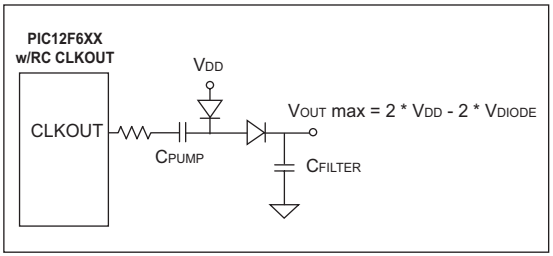
Figure 9-1



- 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

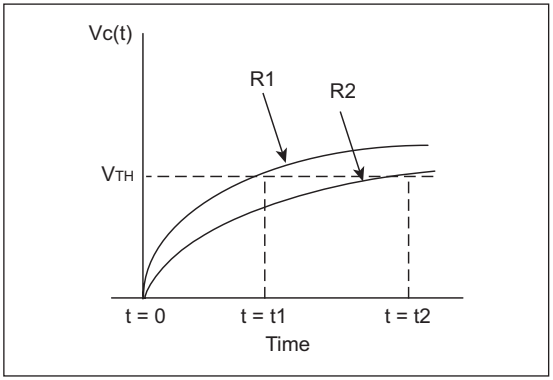
13.3. A/D Method

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

RC Timing Method:

Simple RC step response
$Vc(t) = V_{DD} * (1 - e^{-t/(RC)})$
$t = -RC \ln(1 - V_{TH}/V_{DD})$
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

GENERAL LOW POWER TIPS ‘N TRICKS

TIP #1 Switching Off External Circuits/ Duty Cycle 2-2
TIP #2 Power Budgeting 2-3
TIP #3 Configuring Port Pins 2-4
TIP #4 Use High-Value Pull-Up Resistors..... 2-4
TIP #5 Reduce Operating Voltage 2-4
TIP #6 Use an External Source for CPU Core Voltage 2-5
TIP #7 Battery Backup for PIC MCUs 2-6

DYNAMIC OPERATION TIPS ‘N TRICKS

TIP #8 Enhanced PIC16 Mid-Range Core..... 2-6
TIP #9 Two-Speed Start-Up..... 2-7
TIP #10 Clock Switching 2-7
TIP #11 Use Internal RC Oscillators 2-7
TIP #12 Internal Oscillator Calibration 2-8
TIP #13 Idle and Doze Modes 2-8
TIP #14 Use `NOP` and Idle Mode..... 2-9
TIP #15 Peripheral Module Disable (PMD) Bits 2-9

STATIC POWER REDUCTION TIPS ‘N TRICKS

TIP #16 Deep Sleep Mode..... 2-10
TIP #17 Extended WDT and Deep Sleep WDT 2-10
TIP #18 Low Power Timer1 Oscillator and RTCC..... 2-10
TIP #19 Low Power Timer1 Oscillator Layout.. 2-11
TIP #20 Use LVD to Detect Low Battery..... 2-11
TIP #21 Use Peripheral FIFO and DMA..... 2-11
TIP #22 Ultra Low-Power Wake-Up Peripheral 2-12

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.

TIP #2 Power Budgeting

Power budgeting is a technique that is critical to predicting current consumption and battery life. Power budgeting is performed by calculating the total charge for each mode of operation of an application by multiplying that mode’s current consumption by the time in the mode for a single application loop. The charge for each mode is added, then averaged over the total loop time to get average current. Table 1 calculates a power budget using the application from Figure 2 in Tip #1 using a typical nanoWatt XLP device.

Mode	Time in Mode (mS)	Current (mA)		Charge Current * Time (mA * Sec)
		By Device	Mode Total	
Sleep				
MCU Sleep	1989	0.00005	5.00E-05	9.95E-05
Sensor Off		0		
EEPROM Off		0		
Initialize				
MCU Sleep	1	0.00005	1.66E-02	1.66E-05
Sensor On		0.0165		
EEPROM Off		0		
Sample Sensor				
MCU Run	1	0.048	6.45E-02	6.45E-05
Sensor On		0.0165		
EEPROM Off		0		
Scaling				
MCU Run	1	0.048	4.80E-02	4.80E-05
Sensor Off		0		
EEPROM Off		0		
Storing				
MCU Run	8	0.048	1.05E+00	8.38E-03
Sensor Off		0		
EEPROM On		1		
Total	2000	—	—	8.61E-03

Average Current

$$= \frac{8.61e\text{-}3}{2000e\text{-}3} \ \frac{mA*Sec}{Sec}$$

$$= 0.0043 \text{ mA}$$

Peak Current 1.05 mA

Computing Battery Life

Using the average current from the calculated power budget, it is possible to determine how long a battery will be able to power the application. Table 2 shows lifetimes for typical battery types using the average power from Table 1.

Battery	Capacity (mAh)	Life			
		Hours	Days	Months	Years
CR1212	18	4180	174	5.8	.48
CR1620	75	17417	726	24.2	1.99
CR2032	220	51089	2129	71.0	5.83
Alkaline AAA	1250	290276	12095	403.2	33.14
Alkaline AA	2890	671118	27963	932.1	76.61
Li-ion*	850	197388	8224	274.1	22.53

NOTE: Calculations are based on average current draw only and do not include battery self-discharge.

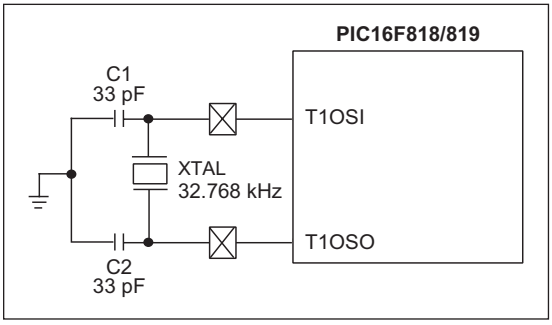
*Varies by size; value used is typical.

After completing a power budget, it is very easy to determine the battery size required to meet the application requirements. If too much power is consumed, it is simple to determine where additional effort needs to be placed to reduce the power consumption.

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 #14 Use NOP and Idle Mode

When waiting on a blocking loop (e.g. waiting for an interrupt), instead put the device into Idle mode to disable the CPU. The peripheral interrupt will wake up the device. Idle mode consumes much less current than constantly reading RAM and jumping back. If the CPU cannot be disabled because the loop required some calculations, such as incrementing a counter, instead of doing a very tight loop that loops many times, add NOPs into the loop. See the code example below. A NOP requires less current to execute than reading RAM or branching operations, so current can be reduced. The overall loop count can be adjusted to account for the extra instructions for the NOPs.

Example:

Replace:

```
while(!_T1IF);
```

with Idle mode:

```
IEC0bits.T1IE = 1;
Idle();
```

and replace:

```
while(!_T1IF){
    i++;
}
```

with extra NOP instructions:

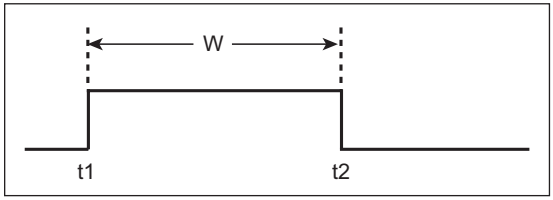
```
while(!_T1IF){
   i++;
    Nop();
    Nop();
    Nop();
    Nop();
    Nop();
}
```

TIP #15 Peripheral Module Disable (PMD) Bits

PIC24, dsPIC DSCs, and PIC32 devices have PMD bits that can be used to disable peripherals that will not be used in the application. Setting these bits disconnects all power to the module as well as SFRs for the module. Because power is completely removed, the PMD bits offer additional power savings over disabling the module by turning off the module’s enable bit. These bits can be dynamically changed so that modules which are only used periodically can be disabled for the remainder of the application. The PMD bits are most effective at high clock speeds and when operating at full speed allowing the average power consumption to be significantly reduced.

TIP #3 Measuring Pulse Width

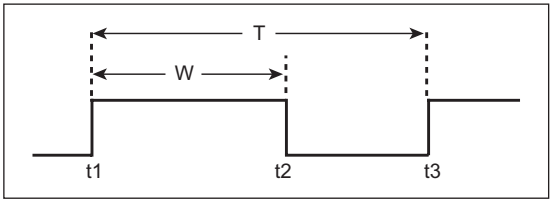
Figure 3-1: Pulse Width



1. Configure control bits CCPxM3:CCPxM0 (CCPxCON<3:0>) to capture every rising edge of the waveform.
2. Configure Timer1 prescaler so that Timer1 will run WMAX without overflowing.
3. Enable the CCP interrupt (CCPxIE bit).
4. When CCP interrupt occurs, save the captured timer value (t1) and reconfigure control bits to capture every falling edge.
5. When CCP interrupt occurs again, subtract saved value (t1) from current captured value (t2) – this result is the pulse width (W).
6. Reconfigure control bits to capture the next rising edge and start process all over again (repeat steps 3 through 6).

TIP #4 Measuring Duty Cycle

Figure 4-1: Duty Cycle



The duty cycle of a waveform is the ratio between the width of a pulse (W) and the period (T). Acceleration sensors, for example, vary the duty cycle of their outputs based on the acceleration acting on a system. The CCP module, configured in Capture mode, can be used to measure the duty cycle of these types of sensors. Here's how:

1. Configure control bits CCPxM3:CCPxM0 (CCPxCON<3:0>) to capture every rising edge of the waveform.
2. Configure Timer1 prescaler so that Timer1 will run TMAX(1) without overflowing.
3. Enable the CCP interrupt (CCPxIE bit).
4. When CCP interrupt occurs, save the captured timer value (t1) and reconfigure control bits to capture every falling edge.

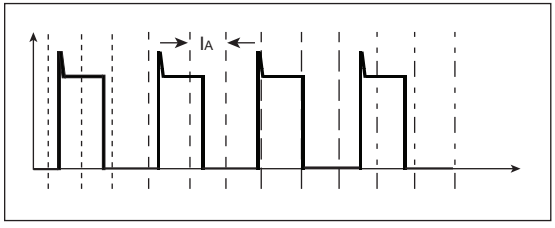
Note 1: TMAX is the maximum pulse period that will occur.

5. When the CCP interrupt occurs again, subtract saved value (t1) from current captured value (t2) – this result is the pulse width (W).
6. Reconfigure control bits to capture the next rising edge.
7. When the CCP interrupt occurs, subtract saved value (t1) from the current captured value (t3) – this is the period (T) of the waveform.
8. Divide T by W – this result is the Duty Cycle.
9. Repeat steps 4 through 8.

TIP #12 Repetitive Phase Shifted Sampling

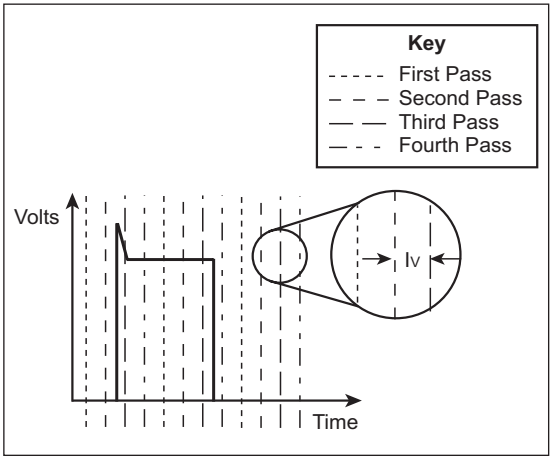
Repetitive phase shifted sampling is a technique to artificially increase the sampling rate of an A/D converter when sampling waveforms that are both periodic and constant from period to period. The technique works by capturing regularly spaced samples of the waveform from the start to finish of the waveform's period. Sampling of the next waveform is then performed in the same manner, except that the start of the sample sequence is delayed a percentage of the sampling period. Subsequent waveforms are also sampled, with each sample sequence slightly delayed from the last, until the delayed start of the sample sequence is equal to one sample period. Interleaving the sample sets then produces a sample set of the waveform at a higher sample rate. Figure 12-1 shows an example of a high frequency waveform.

Figure 12-1: High Frequency Periodic Waveform



As indicated in the key, the finely dotted lines show where the A/D readings are taken during the first period of the waveform. The medium sized dashed lines show when the A/D readings are taken during the second period, and so on. Figure 12-2 shows these readings transposed onto one period.

Figure 12-2: Transposed Waveform



The CCP module is configured in Compare Special Event Trigger mode to accomplish this task. The phase shift is implemented by picking values of CCPRxL and CCPRxH that are not synchronous with the period of the sampling waveform. For instance, if the period of a waveform is 100 μs, then sampling at a rate of once every 22 μs will give the following set of sample times over 11 periods (all values in μs).

1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	11th
0	10	20	8	18	6	16	4	14	2	12
22	32	42	30	40	28	38	26	36	24	34
44	54	64	52	62	50	60	48	58	46	56
66	76	86	74	84	72	82	70	80	68	78
88	98		96		94		92		90	

When these numbers are placed in sequential order, they reveal a virtual sampling interval (I_V) of 2 μs from 0 μs to 100 μs, although the actual sampling interval (I_A) is 22 μs.

TIP #2 Faster Code for Detecting Change

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

Example 2-1

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

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

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

Example 2-2

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

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

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

Example 2-3

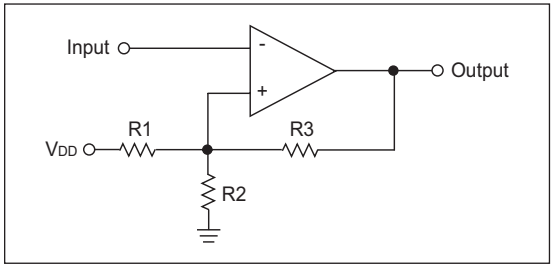
```
Test
   BTFSS CMCON,COUT ;test Cout
   RETLW 0          ;if clear, return 0
   MOVLW CINVBIT    ;if set, invert Cout
   XORWF CMCON,f    ;forces Cout to 0
   BTFSS CMCON,CINV ;test Cout polarity
   RETLW CHNGFLAG   ;if clear, return
                    ;CHNGFLAG
   RETLW COUTMASK + CHNGFLAG;if set,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

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

TIP #3 Hysteresis

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

Figure 3-1: Comparator with Hysteresis



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

Equation 3-1

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

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

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

Equation 3-2

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

Equation 3-3

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

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

Equation 3-4

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

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

Equation 3-5

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

Example:

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

CHAPTER 5
DC Motor Control Tips ‘n Tricks

Table Of Contents

TIPS ‘N TRICKS INTRODUCTION

TIP #1: Brushed DC Motor Drive Circuits 5-2
TIP #2: Brushless DC Motor Drive Circuits..... 5-3
TIP #3: Stepper Motor Drive Circuits 5-4
TIP #4: Drive Software..... 5-6
TIP #5: Writing a PWM Value to the CCP Registers with a Mid-Range PIC® MCU..... 5-7
TIP #6: Current Sensing 5-8
TIP #7: Position/Speed Sensing 5-9

Application Note References.....5-11

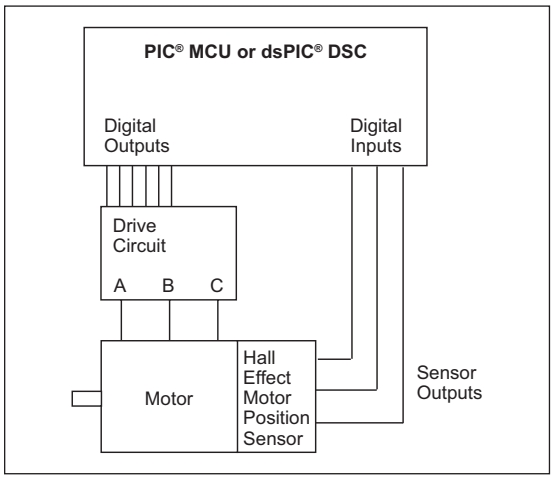
Motor Control Development Tools 5-11

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).

Figure 2-3: Quadrature Decoder (Sensor Motor)



Application notes describing Brushless DC Motor Control are listed below and can be found on the Microchip web site at: www.microchip.com.

- AN857, “*Brushless DC Motor Control Made Easy*” (DS00857)
- AN885, “*Brushless DC Motor Fundamentals*” (DS00885)
- AN899, “*Brushless DC Motor Control Using PIC18FXX31*” (DS00899)
- AN901, “*Using the dsPIC30F for Sensorless BLDC Control*” (DS00901)
- AN957, “*Sensored BLDC Motor Control Using dsPIC30F2010*” (DS00957)
- AN992, “*Sensorless BLDC Motor Control Using dsPIC30F2010*” (DS00992)
- AN1017, “*Sinusoidal Control of PMSM with dsPIC30F DSC*” (DS01017)
- GS005, “*Using the dsPIC30F Sensorless Motor Tuning Interface*” (DS93005)

TIP #3 Stepper Motor Drive Circuits

Stepper motors are similar to Brushless DC motors in that the control system must commutate the motor through the entire rotation cycle. Unlike the brushless motor, the position and speed of a stepping motor is predictable and does not require the use of sensors.

There are two basic types of stepper motors, although some motors are built to be used in either mode. The simplest stepper motor is the unipolar motor. This motor has four drive connections and one or two center tap wires that are tied to ground or Vsupply, depending on the implementation. Other motor types are the bipolar stepper and various combinations of unipolar and bipolar, as shown in Figure 3-1 and Figure 3-2. When each drive connection is energized, one coil is driven and the motor rotates one step. The process is repeated until all the windings have been energized. To increase the step rate, often the voltage is increased beyond the motors rated voltage. If the voltage is increased, some method of preventing an over current situation is required.

There are many ways to control the winding current, but the most popular is a chopper system that turns off current when it reaches an upper limit and enables the current flow a short time later. Current sensor systems are discussed in Tip #6. Some systems are built with a current chopper, but they do not detect the current, rather the system is designed to begin a fixed period chopping cycle after the motor has stepped to the next position. These are simpler systems to build, as they only require a change in the software.

CHAPTER 7
Intelligent Power Supply Design Tips ‘n Tricks

Table Of Contents

TIPS ‘N TRICKS INTRODUCTION

TIP #1: Soft-Start Using a PIC10F200 7-2
TIP #2: A Start-Up Sequencer 7-3
TIP #3: A Tracking and Proportional Soft-Start of Two Power Supplies..... 7-4
TIP #4: Creating a Dithered PWM Clock 7-5
TIP #5: Using a PIC® Microcontroller as a Clock Source for a SMPS PWM Generator 7-6
TIP #6: Current Limiting Using the MCP1630 7-7
TIP #7: Using a PIC® Microcontroller for Power Factor Correction 7-8
TIP #8: Transformerless Power Supplies..... 7-9
TIP #9: An IR Remote Control Actuated AC Switch for Linear Power Supply Designs 7-10
TIP #10: Driving High Side FETs 7-11
TIP #11: Generating a Reference Voltage with a PWM Output 7-12
TIP #12: Using Auto-Shutdown CCP 7-13
TIP #13: Generating a Two-Phase Control Signal..... 7-14
TIP #14: Brushless DC Fan Speed Control 7-15
TIP #15: High Current Delta-Sigma Based Current Measurement Using a Slotted Ferrite and Hall Effect Device 7-16
TIP #16: Implementing a PID Feedback Control in a PIC12F683-Based SMPS Design 7-17
TIP #17: An Error Detection and Restart Controller..... 7-18
TIP #18: Data-Indexed Software State Machine 7-19
TIP #19: Execution Indexed Software State Machine 7-20
TIP #20: Compensating Sensors Digitally 7-21
TIP #21: Using Output Voltage Monitoring to Create a Self-Calibration Function 7-22

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.

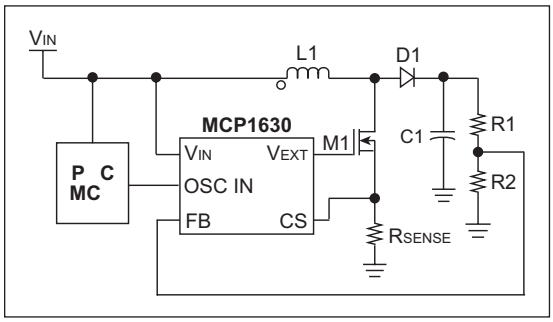
This document showcases several examples in which a PIC microcontroller may be used to increase the functionality of a design with a minimal increase in cost.

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 #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.

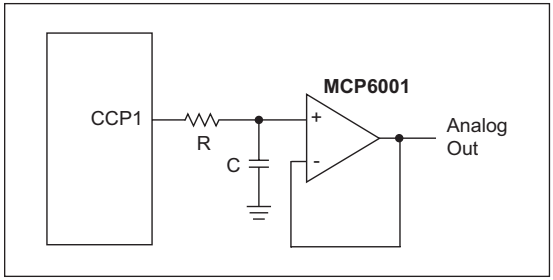
The switching frequency of the MCP1630 can be adjusted by changing the frequency of the clock source. The maximum on-timer of the MCP1630 PWM can be adjusted by changing the duty cycle of the clock source.

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 #11 Generating a Reference Voltage with a PWM Output

Figure 11-1: Low-Pass Filter



A PWM signal can be used to create a Digital-to-Analog Converter (DAC) 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, the PWM frequency (F_{PWM}) should be significantly higher than the bandwidth (F_{BW}) of the desired analog signal. Equation 11-1 shows this relation.

Equation 11-1

$$F_{PWM} = K \bullet F_{BW}$$

Where harmonics decrease as K increases.

R and C are chosen based on the following equation:

Equation 11-2

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

Where harmonics decrease as K increases.

Choose the R value based on drive capability and then calculate the required C value. The attenuation of the PWM frequency for a given RC filter is shown in Equation 11-3.

Equation 11-3

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

If the attenuation calculated in Equation 11-3 is not sufficient, then K must be increased in Equation 11-1.

In order to sufficiently attenuate the harmonics, it may be necessary to use small capacitor values or large resistor values. Any current draw will effect the voltage across the capacitor. Adding an op amp allows the analog voltage to be buffered and, because of this, any current drawn will be supplied by the op amp and not the filter capacitor.

For more information on using a PWM signal to generate an analog output, refer to AN538, “*Using PWM to Generate Analog Output*” (DS00538).

TIP #14 Brushless DC Fan Speed Control

There are several methods to control the speed of a DC brushless fan. The type of fan, allowable power consumption and the type of control desired are all factors in choosing the appropriate type.

Figure 14-1: Low-Side PWM Drive

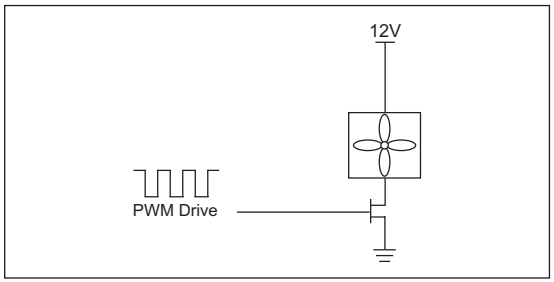
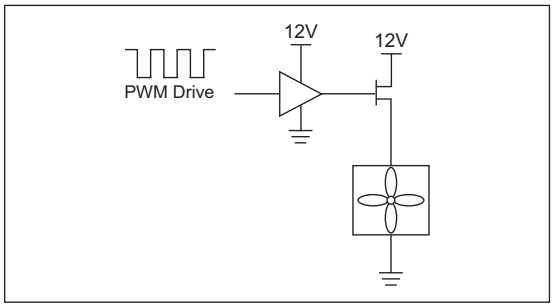


Figure 14-2: High-Side PWM Drive



Method 1 – Pulse-Width Modulation

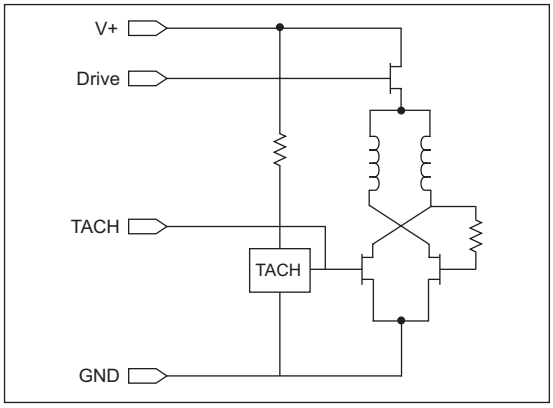
As shown in Figure 14-1 and Figure 14-2, a simple PWM drive may be used to switch a two-wire fan on and off. While it is possible to use the circuit in Figure 14-1 without a high-side MOSFET driver, some manufacturers state that switching on the low side of the fan will void the warranty.

Because of this, it is necessary to switch the high side of the fan in order to control the speed. The simplest type of speed control is ‘on’ or ‘off’. However, if a higher degree of control is desired, PWM can be used to vary the speed of the fan.

For 3-wire fans, the tachometer output will not be accurate if PWM is used. The sensor providing the tachometer output on 3-wire fans is powered from the same supply as the fan coils, thus using a PWM to control fan speed will render the fan’s tachometer inaccurate.

One solution is to use a 4-wire fan which includes both the tachometer output and a drive input. Figure 14-3 shows a diagram of a 4-wire fan.

Figure 14-3: Typical 4-Wire Fan



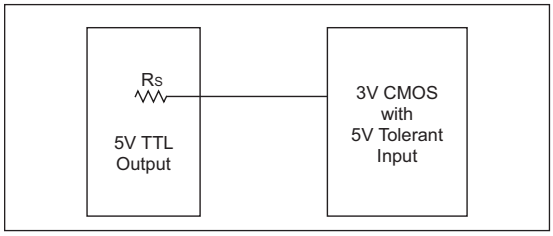
A 4-wire fan allows speed to be controlled using PWM via the Drive line. Since power to the tachometer sensor is not interrupted, it will continue to output the correct speed.

TIP #9 5V → 3.3V Direct Connect

5V outputs have a typical V_{OH} of 4.7 volts and a V_{OL} of 0.4 volts and a 3.3V LVCMOS input will have a typical V_{IH} of 0.7 x V_{DD} and a V_{IL} of 0.2 x V_{DD}.

When the 5V output is driving low, there are no problems because the 0.4 volt output is less than in the input threshold of 0.8 volts. When the 5V output is high, the V_{OH} of 4.7 volts is greater than 2.1 volt V_{IH}, therefore, we can directly connect the 2 pins with no conflicts if the 3.3V CMOS input is 5 volt tolerant.

Figure 9-1: 5V Tolerant Input



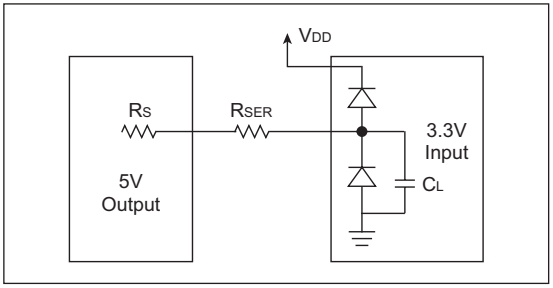
If the 3.3V CMOS input is not 5 volt tolerant, then there will be an issue because the maximum volt specification of the input will be exceeded.

See Tips 10-13 for possible solutions.

TIP #10 5V → 3.3V With Diode Clamp

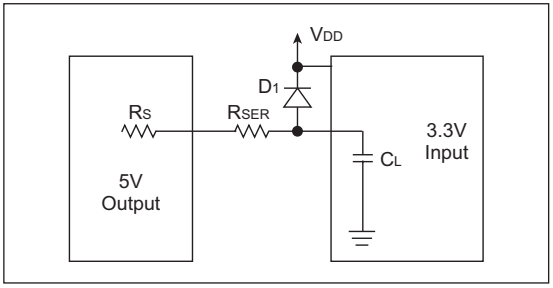
Many manufacturers protect their I/O pins from exceeding the maximum allowable voltage specification by using clamping diodes. These clamping diodes keep the pin from going more than a diode drop below V_{SS} and a diode drop above V_{DD}. To use the clamping diode to protect the input, you still need to look at the current through the clamping diode. The current through the clamp diodes should be kept small (in the micro amp range). If the current through the clamping diodes gets too large, then you risk the part latching up. Since the source resistance of a 5V output is typically around 10Ω, an additional series resistor is still needed to limit the current through the clamping diode as shown Figure 10-1. The consequence of using the series resistor is it will reduce the speed at which we can switch the input because the RC time constant formed the capacitance of the pin (C_L).

Figure 10-1: Clamping Diodes on the Input



If the clamping diodes are not present, a single external diode can be added to the circuit as shown in Figure 10-2.

Figure 10-2: Without Clamping Diodes



TIP #16 5V → 3.3V Active Analog Attenuator

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

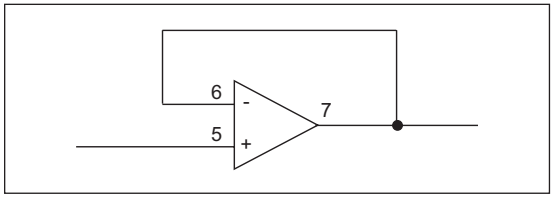
The simplest method of converting a 5V analog signal to a 3.3V analog signal is to use a resistor divider with a ratio R1:R2 of 1.7:3.3. However, there are a few problems with this.

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).

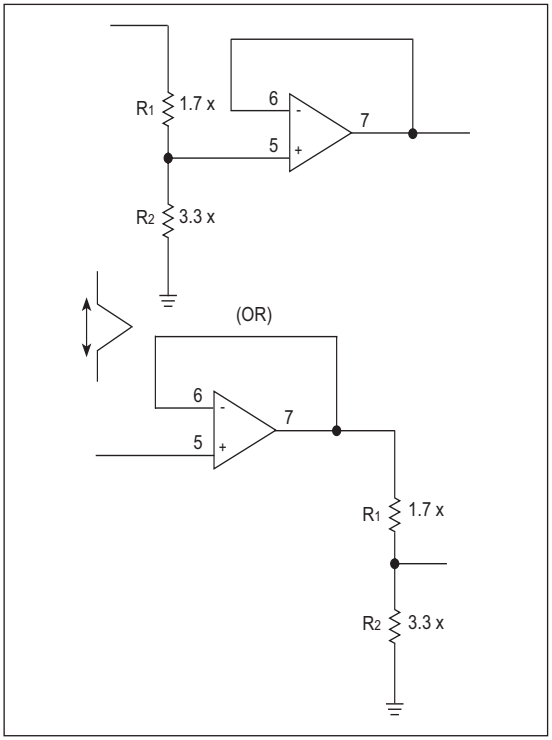
Figure 16-1: Unity Gain



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.

Table 18-1: Bipolar Transistor DC Specifications

Characteristic	Sym	Min	Max	Unit	Test Condition
OFF CHARACTERISTICS					
Collector-Base Breakdown Voltage	$V_{(BR)CBO}$	60	–	V	I_C = 50 µA, I_E = 0
Collector-Emitter Breakdown Voltage	$V_{(BR)CEO}$	50	–	V	I_C = 1.0 mA, I_B = 0
Emitter-Base Breakdown Voltage	$V_{(BR)EBO}$	7.0	–	V	I_E = 50 µA, I_C = 0
Collector Cutoff Current	I_{CBO}	–	100	nA	V_{CB} = 60V
Emitter Cutoff Current	I_{EBO}	–	100	nA	V_{EB} = 7.0V
ON CHARACTERISTICS					
DC Current Gain	h_{FE}	120 180 270	270 390 560	–	V_{CE} = 6.0V, I_C = 1.0 mA
Collector-Emitter Saturation Voltage	$V_{CE(SAT)}$	–	0.4	V	I_C = 50 mA, I_B = 5.0 mA

When using bipolar transistors as switches to turn on and off loads controlled by the microcontroller I/O port pin, use the minimum h_{FE} specification and margin to ensure complete device saturation.

Equation 18-1: Calculating the Base Resistor Value

$$R_{BASE} = \frac{(V_{DD} - V_{BE}) \times h_{FE} \times R_{LOAD}}{V_{LOAD}}$$

3V Technology Example

V_{DD} = +3V, V_{LOAD} = +40V, R_{LOAD} = 400Ω, h_{FE} min. = 180, V_{BE} = 0.7V

R_{BASE} = 4.14 kΩ, I/O port current = 556 µA

5V Technology Example

V_{DD} = +5V, V_{LOAD} = +40V, R_{LOAD} = 400Ω, h_{FE} min. = 180, V_{BE} = 0.7V

R_{BASE} = 7.74 kΩ, I/O port current = 556 µA

For both examples, it is good practice to increase base current for margin. Driving the base with 1 mA to 2 mA would ensure saturation at the expense of increasing the input power consumption.