



Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!



## Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832

Email & Skype: info@chipsmall.com Web: www.chipsmall.com

Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China





# Compiled Tips ‘N Tricks Guide

---

**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.

---

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

**Trademarks**

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, rPIC, SmartShunt and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


FilterLab, Hampshire, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, nanoWatt XLP, PICKit, PICDEM, PICDEM.net, PICtail, PIC<sup>32</sup> logo, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rLAB, Select Mode, Total Endurance, TSHARC, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2009, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

**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

### 8-pin Flash PIC® Microcontrollers Tips 'n Tricks

#### 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:	V <sub>DD</sub> 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

### PIC® Microcontroller Low Power Tips 'n Tricks

#### 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

### PIC® Microcontroller CCP and ECCP Tips 'n Tricks

#### CAPTURE TIPS 'N TRICKS

TIP #1:	Measuring the Period of a Square Wave ...	3-3
TIP #2:	Measuring the Period of a Square Wave with Averaging .....	3-3
TIP #3:	Measuring Pulse Width .....	3-4
TIP #4:	Measuring Duty Cycle .....	3-4
TIP #5:	Measuring RPM Using an Encoder.....	3-5
TIP #6:	Measuring the Period of an Analog Signal .	3-6

#### COMPARE TIPS 'N TRICKS

TIP #7:	Periodic Interrupts .....	3-8
TIP #8:	Modulation Formats.....	3-9
TIP #9:	Generating the Time Tick for a RTOS .....	3-10
TIP #10:	16-Bit Resolution PWM .....	3-10
TIP #11:	Sequential ADC Reader .....	3-11
TIP #12:	Repetitive Phase Shifted Sampling.....	3-12

#### PWM TIPS 'N TRICKS

TIP #13:	Deciding on PWM Frequency.....	3-14
TIP #14:	Unidirectional Brushed DC Motor Control Using CCP .....	3-14
TIP #15:	Bidirectional Brushed DC Motor Control Using ECCP .....	3-15
TIP #16:	Generating an Analog Output.....	3-16
TIP #17:	Boost Power Supply .....	3-17
TIP #18:	Varying LED Intensity .....	3-18
TIP #19:	Generating X-10 Carrier Frequency.....	3-18

#### COMBINATION CAPTURE AND COMPARE TIPS

TIP #20:	RS-232 Auto-baud .....	3-19
TIP #21:	Dual-Slope Analog-to-Digital Converter .....	3-21

# Tips 'n Tricks Table of Contents

## **PIC® Microcontroller Comparator**

### **Tips 'n Tricks**

TIP #1:	Low Battery Detection .....	4-2
TIP #2:	Faster Code for Detecting Change.....	4-3
TIP #3:	Hysteresis.....	4-4
TIP #4:	Pulse Width Measurement .....	4-5
TIP #5:	Window Comparison .....	4-6
TIP #6:	Data Slicer.....	4-7
TIP #7:	One-Shot.....	4-8
TIP #8:	Multi-Vibrator (Square Wave Output).....	4-9
TIP #9:	Multi-Vibrator (Ramp Wave Output).....	4-10
TIP #10:	Capacitive Voltage Doubler .....	4-11
TIP #11:	PWM Generator .....	4-12
TIP #12:	Making an Op Amp Out of a Comparator...4-13	
TIP #13:	PWM High-Current Driver .....	4-14
TIP #14:	Delta-Sigma ADC .....	4-15
TIP #15:	Level Shifter .....	4-16
TIP #16:	Logic: Inverter.....	4-16
TIP #17:	Logic: AND/NAND Gate .....	4-17
TIP #18:	Logic: OR/NOR Gate.....	4-18
TIP #19:	Logic: XOR/XNOR Gate.....	4-19
TIP #20:	Logic: Set/Reset Flip Flop.....	4-20

## **PIC® Microcontroller DC Motor Control**

### **Tips 'n Tricks**

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

## **LCD PIC® Microcontroller Tips 'n Tricks**

TIP #1:	Typical Ordering Considerations and Procedures for Custom Liquid Displays ....	6-2
TIP #2:	LCD PIC® MCU Segment/Pixel Table.....	6-2
TIP #3:	Resistor Ladder for Low Current.....	6-3
TIP #4:	Contrast Control with a Buck Regulator ....	6-5
TIP #5:	Contrast Control Using a Boost Regulator.....	6-5
TIP #6:	Software Controlled Contrast with PWM for LCD Contrast Control.....	6-6
TIP #7:	Driving Common Backlights .....	6-7
TIP #8:	In-Circuit Debug (ICD).....	6-8
TIP #9:	LCD in Sleep Mode .....	6-8
TIP #10:	How to Update LCD Data Through Firmware .....	6-9
TIP #11:	Blinking LCD.....	6-9
TIP #12:	4 x 4 Keypad Interface that Conserves Pins for LCD Segment Drivers .....	6-10
Application Note References .....		6-11

## **Intelligent Power Supply Design**

### **Tips 'n Tricks**

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

## **3V Tips 'n Tricks**

TIP #1:	Powering 3.3V Systems From 5V Using an LDO Regulator .....	8-3
TIP #2:	Low-Cost Alternative Power System Using a Zener Diode .....	8-4
TIP #3:	Lower Cost Alternative Power System Using 3 Rectifier Diodes.....	8-4
TIP #4:	Powering 3.3V Systems From 5V Using Switching Regulators .....	8-5
TIP #5:	3.3V → 5V Direct Connect.....	8-6
TIP #6:	3.3V → 5V Using a MOSFET Translator....	8-6
TIP #7:	3.3V → 5V Using A Diode Offset.....	8-7
TIP #8:	3.3V → 5V Using A Voltage Comparator....	8-8
TIP #9:	5V → 3.3V Direct Connect.....	8-9
TIP #10:	5V → 3.3V With Diode Clamp.....	8-9
TIP #11:	5V → 3.3V Active Clamp .....	8-10
TIP #12:	5V → 3.3V Resistor Divider.....	8-10
TIP #13:	3.3V → 5V Level Translators.....	8-12
TIP #14:	3.3V → 5V Analog Gain Block.....	8-13
TIP #15:	3.3V → 5V Analog Offset Block.....	8-13
TIP #16:	5V → 3.3V Active Analog Attenuator.....	8-14
TIP #17:	5V → 3V Analog Limiter .....	8-15
TIP #18:	Driving Bipolar Transistors .....	8-16
TIP #19:	Driving N-Channel MOSFET Transistors ...8-18	

# CHAPTER 1

## 8-Pin Flash PIC<sup>®</sup> 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:	V <sub>DD</sub> Self Starting Circuit.....	1-7
TIP #12:	Using PIC <sup>®</sup> 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<sup>®</sup> microcontrollers (MCU) are used in a wide range of everyday products, from toothbrushes, hair dryers and rice cookers to industrial, automotive and medical products.

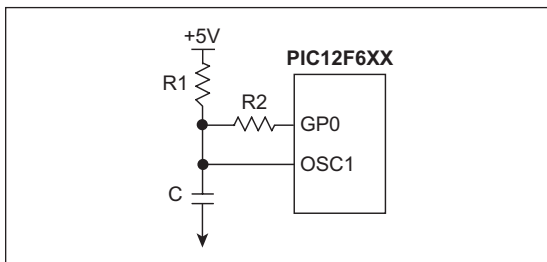
The PIC12F629/675 MCUs merge all the advantages of the PIC MCU architecture and the flexibility of Flash program memory into an 8-pin package. They provide the features and intelligence not previously available due to cost and board space limitations. Features include a 14-bit instruction set, small 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<sup>™</sup> and MPLAB<sup>®</sup> 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 #1 Dual Speed RC Oscillator

Figure 1-1



1. After reset I/O pin is High-Z
2. Output ‘1’ on I/O pin
3. R1, R2 and C determine OSC frequency
4. Also works with additional capacitors

Frequency of PIC MCU in external RC oscillator mode depends on resistance and capacitance on OSC1 pin. Resistance is changed by the output voltage on GP0. GP0 output ‘1’ puts R2 in parallel with R1 reduces OSC1 resistance and increases OSC1 frequency. GP0 as an input increases the OSC1 resistance by minimizing current flow through R2, and decreases frequency and power consumption.

### Summary:

- GP0 = Input: Slow speed for low current
- GP0 = Output high: High speed for fast processing

## TIP #2 Input/Output Multiplexing

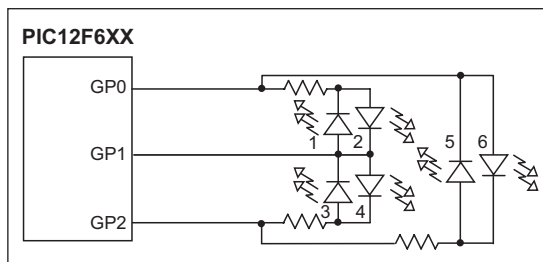
Individual diodes and some combination of diodes can be enabled by driving I/Os high and low or switching to inputs (Z). The number of diodes (D) that can be controlled depends on the number of I/Os (GP) used.

The equation is:  $D = GP \times (GP - 1)$ .

### Example 2-1: Six LEDs on Three I/O Pins

GPx	LEDs
0 1 2	1 2 3 4 5 6
0 0 0	0 0 0 0 0 0
0 1 Z	1 0 0 0 0 0
1 0 Z	0 1 0 0 0 0
Z 0 1	0 0 1 0 0 0
Z 1 0	0 0 0 1 0 0
0 Z 1	0 0 0 0 1 0
1 Z 0	0 0 0 0 0 1
0 0 1	0 0 1 0 1 0
0 1 0	1 0 0 1 0 0
0 1 1	1 0 0 0 1 0
1 0 0	0 1 0 0 0 1
1 0 1	0 1 1 0 0 0
1 1 0	0 0 0 1 0 1
1 1 1	0 0 0 0 0 0

Figure 2-1



### 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

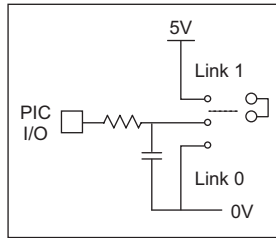
To check state 0:

- 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

```

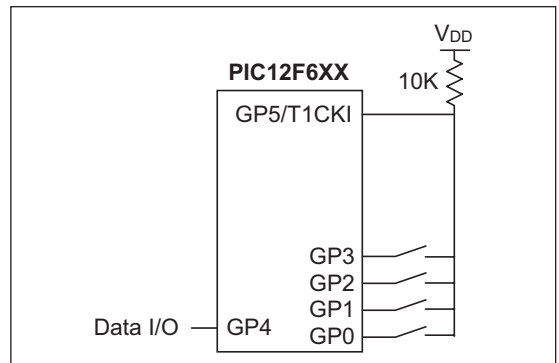
movlw    b'11111111'
movwf   TRISIO
DIP
movlw    b'00000111'
movwf   T1CON
movlw    b'11111110'
movwf   Mask
clrfs   GPIO

LOOP
clrfs   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





### TIP #5 Scanning Many Keys With One Input

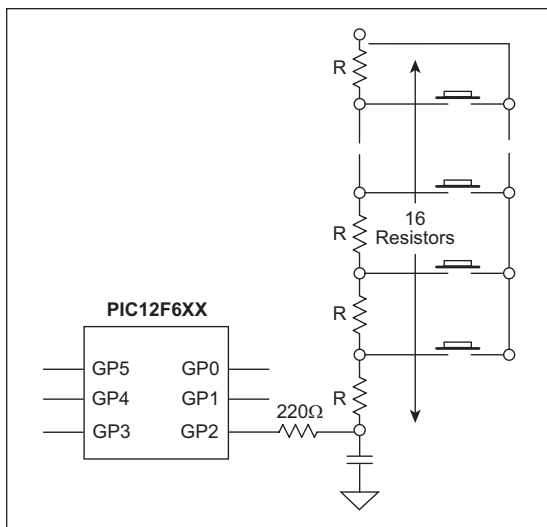
The time required to charge a capacitor depends on resistance between  $V_{DD}$  and capacitor. When a button is pressed,  $V_{DD}$  is supplied to a different point in the resistor ladder. The resistance between  $V_{DD}$  and the capacitor is reduced, which reduces the charge time of the capacitor. A timer is used with a comparator or changing digital input to measure the capacitor charge time. The charge time is used to determine which button is pressed.

Software sequence:

1. Configure GP2 to output a low voltage to discharge capacitor through I/O resistor.
2. Configure GP2 as one comparator input and  $CV_{REF}$  as the other.
3. Use a timer to measure when the comparator trips. If the time measured is greater than the maximum allowed time, then repeat; otherwise determine which button is pressed.

When a key is pressed, the voltage divider network changes the RC ramp rate.

Figure 5-1



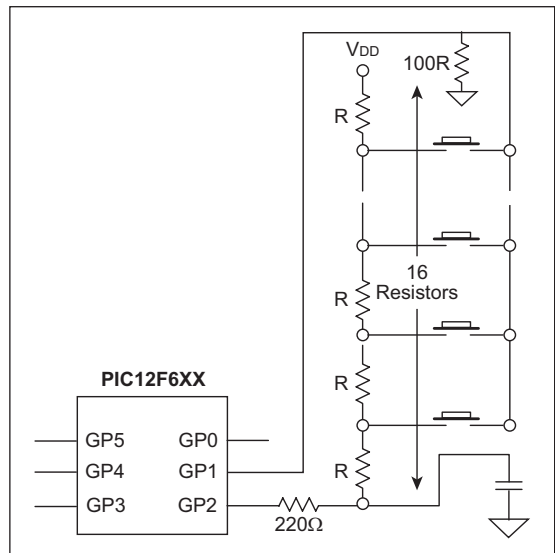
See AN512, "Implementing Ohmmeter/ Temperature Sensor" for code ideas.

### TIP #6 Scanning Many Keys and Wake-up From Sleep

An additional I/O can be added to wake the part when a button is pressed. Prior to Sleep, configure GP1 as an input with interrupt-on-change enabled and GP2 to output high. The pull-down resistor holds GP1 low until a button is pressed. GP1 is then pulled high via GP2 and  $V_{DD}$  generating an interrupt. After wake-up, GP2 is configured to output low to discharge the capacitor through the  $220\Omega$  resistor. GP1 is set to output high and GP2 is set to an input to measure the capacitor charge time.

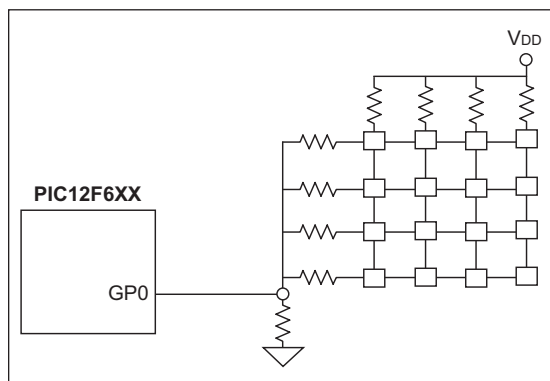
- GP1 pin connected to key common
- Enable wake-up on port change
- Set GP1 as input and GP2 high prior to Sleep
- If key is pressed the PIC MCU wakes up, GP2 must be set low to discharge capacitor
- Set GP1 high upon wake-up to scan keystroke

Figure 6-1

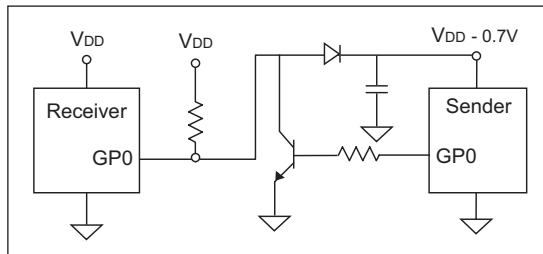


**TIP #7 4x4 Keyboard with 1 Input**

By carefully selecting the resistor values, each button generates a unique voltage. This voltage is measured by the A/D to determine which button is pressed. Higher precision resistors should be used to maximize voltage uniqueness. The A/D will read near 0 when no buttons are pressed.

**Figure 7-1****TIP #8 One Pin Power/Data**

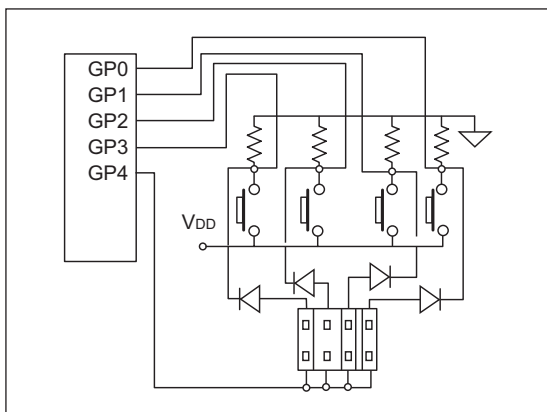
A single I/O can be used for both a single-direction communication and the power source for another microcontroller. The I/O line is held high by the pull-up resistor connected to  $V_{DD}$ . The sender uses a pull-down transistor to pull the data line low or disables the transistor to allow the pull-up to raise it to send data to the receiver.  $V_{DD}$  is supplied to the sender through the data line. The capacitor stabilizes the sender's  $V_{DD}$  and a diode prevents the capacitor from discharging through the I/O line while it is low. Note that the  $V_{DD}$  of the sender is a diode-drop lower than the receiver.

**Figure 8-1**

## 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.

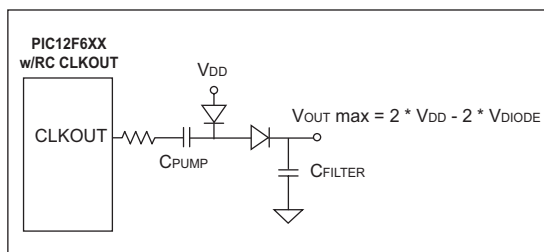
Figure 9-1



- 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.

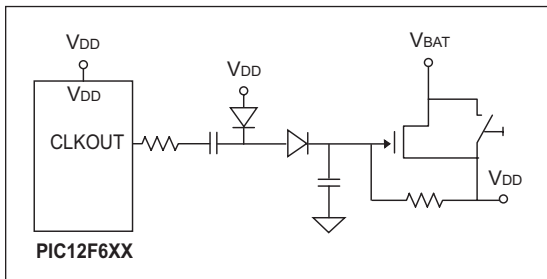
### TIP #11 V<sub>DD</sub> Self Starting Circuit

Building on the previous topic, the same charge pump can be used by the MCU to supply its own V<sub>DD</sub>. Before the switch is pressed, V<sub>BAT</sub> has power and the V<sub>DD</sub> points are connected together but unpowered. When the button is pressed, power is supplied to V<sub>DD</sub> and the MCUs CLKOUT (in external RC oscillator mode) begins toggle. The voltage generated by the charge pump turns on the FET allowing V<sub>DD</sub> to remain powered. To power down the MCU, execute a Sleep instruction. This allows the MCU to switch off its power source via software.

#### Advantages:

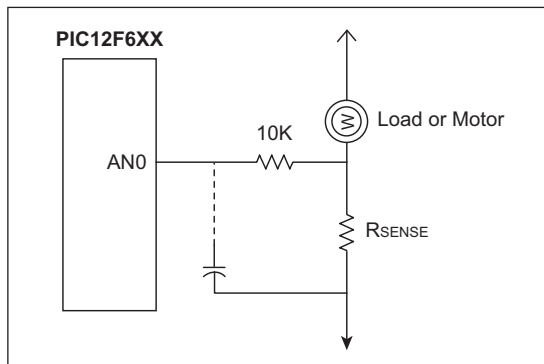
- PIC MCU leakage current nearly 0
- Low cost (uses n-channel FET)
- Reliable
- No additional I/O pins required

Figure 11-1



### TIP #12 Using PIC® MCU A/D For Smart Current Limiter

Figure 12-1



- Detect current through low side sense resistor
- Optional peak filter capacitor
- Varying levels of overcurrent response can be realized in software

By adding a resistor (R<sub>SENSE</sub>) in series with a motor, the A/D can be used to measure in-rush current, provide current limiting, over-current recovery or work as a smart circuit breaker. The 10K resistor limits the analog channel current and does not violate the source impedance limit of the A/D.

## 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.

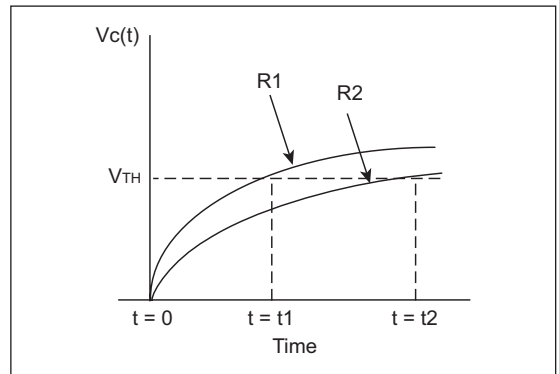
- 13.1. RC Timing Method (with reference resistor)
- 13.2. Charge Balancing Method
- 13.3. A/D Method

## 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.

**Application Notes:**

AN512, "Implementing Ohmmeter/Temperature Sensor"

AN611, "Resistance and Capacitance Meter Using a PIC16C622"

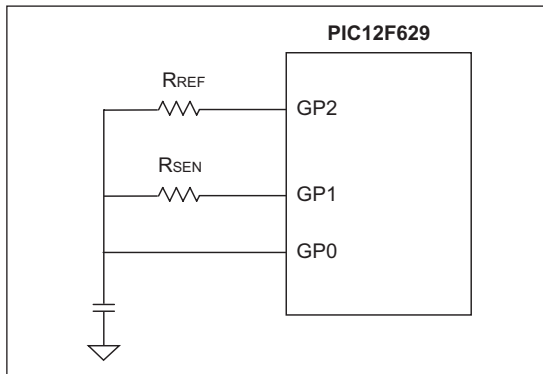
Here is the schematic and software flow for using a reference resistor to improve the accuracy of an analog sensor reading. The reference resistor ( $R_{REF}$ ) and sensor ( $R_{SEN}$ ) are assigned an I/O and share a common capacitor. GP0 is used to discharge the capacitor and represents the capacitor voltage.

Through software, a timer is used to measure when GP0 switches from a '0' to a '1' for the sensor and reference measurements. Any difference measured between the reference measurement and its calibrated measurement is used to adjust the sensor reading, resulting in a more accurate measurement.

The comparator and comparator reference on the PIC12F629/675 can be used instead of a port pin for a more accurate measurement. Polypropylene capacitors are very stable and beneficial in this type of application.

1. Set GP1 and GP2 to inputs, and GP0 to a low output to discharge C
2. Set GP0 to an input and GP1 to a high output
3. Measure  $t_{R_{SEN}}$  (GP0 changes to 1)
4. Repeat step 1
5. Set GP0 to an input and GP2 to a high output
6. Measure  $t_{R_{REF}}$  (GP0 changes to 1)
7. Use film polypropylene capacitor
8.  $R_{TH} = x R_{REF} \frac{t_{R_{SEN}}}{t_{R_{REF}}}$

**Figure 13-2**



Other alternatives: voltage comparator in the PIC12F6XX to measure capacitor voltage on GP0.

### Tip #13.2 Reading a Sensor With Higher Accuracy – Charge Balancing Method

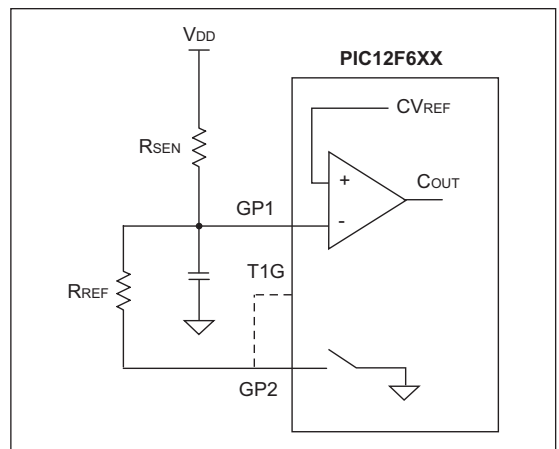
1. Sensor charges a capacitor
2. Reference resistor discharges the capacitor
3. Modulate reference resistor to maintain constant average charge in the capacitor
4. Use comparator to determine modulation

To improve resolution beyond 10 or 12 bits, a technique called “Charge Balancing” can be used. The basic concept is for the MCU to maintain a constant voltage on a capacitor by either allowing the charge to build through a sensor or discharge through a reference resistor. A timer is used to sample the capacitor voltage on regular intervals until a predetermined number of samples are counted. By counting the number of times the capacitor voltage is over an arbitrary threshold, the sensor voltage is determined.

The comparator and comparator voltage reference ( $CV_{REF}$ ) on the PIC12F629/675 are ideal for this application.

1. GP1 average voltage =  $CV_{REF}$
2. Time base as sampling rate
3. At the end of each time base period:
  - If  $GP1 > CV_{REF}$ , then GP2 Output Low
  - If  $GP1 < CV_{REF}$ , then GP2 Input mode
4. Accumulate the GP2 lows over many samples
5. Number of samples determines resolution
6. Number of GP2 lows determine effective duty cycle of  $R_{REF}$

Figure 13-3



**Tip #13.3 Reading a Sensor With Higher Accuracy – A/D Method**

NTC (Negative Temperature Coefficient) sensors have a non-linear response to temperature changes. As the temperature drops, the amount the resistance changes becomes less and less. Such sensors have a limited useful range because the resolution becomes smaller than the A/D resolution as the temperature drops. By changing the voltage divider of the  $R_{SEN}$ , the temperature range can be expanded.

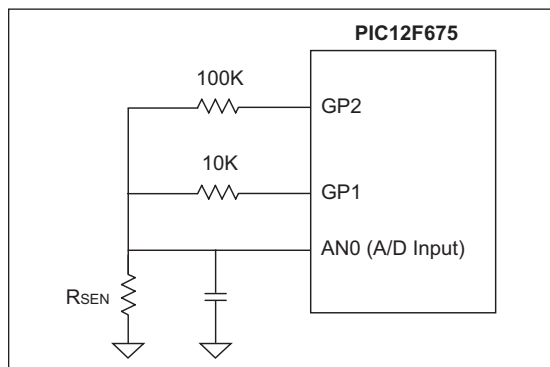
To select the higher temperature range, GP1 outputs '1' and GP2 is set as an input. For the lower range, GP2 outputs '1' and GP1 is configured as an input. The lower range will increase the amount the sensor voltage changes as the temperature drops to allow a larger usable sensor range.

**Summary:**

High range: GP1 output '1' and GP2 input  
 Low range: GP1 input and GP2 output '1'

1. 10K and 100K resistors are used to set the range
2.  $V_{REF}$  for A/D =  $V_{DD}$
3.  $R_{th}$  calculation is independent of  $V_{DD}$
4.  $Count = R_{SEN} / (R_{SEN} + R_{REF}) \times 255$
5. Don't forget to allow acquisition time for the A/D

**Figure 13-4**

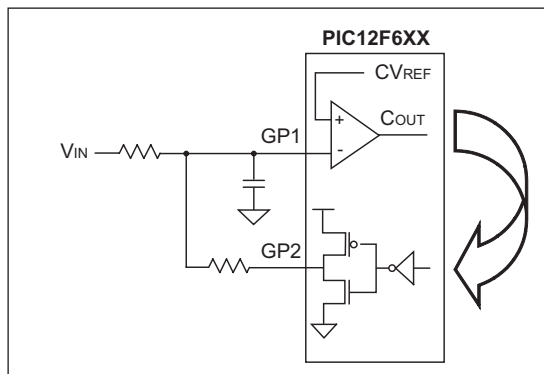


**TIP #14 Delta-Sigma Converter**

The charge on the capacitor on GP1 is maintained about equal to the  $CV_{REF}$  by the MCU monitoring  $C_{OUT}$  and switching GP2 from Input mode or output low appropriately. A timer is used to sample the  $C_{OUT}$  bit on a periodic basis. Each time GP2 is driven low, a counter is incremented. This counter value corresponds to the input voltage.

To minimize the affects of external component tolerances, temperature, etc., the circuit can be calibrated. Apply a known voltage to the input and allow the microcontroller to count samples until the expected result is calculated. By taking the same number of samples for subsequent measurements, they become calibrated measurements.

**Figure 14-1**



1. GP1 average voltage =  $CV_{REF}$
2. Time base as sampling rate
3. At the end of each time base period:
  - If  $GP1 > CV_{REF}$ , then GP2 Output Low
  - If  $GP1 < CV_{REF}$ , then GP2 Output High
4. Accumulate the GP2 lows over many samples
5. Number of samples determines resolution



### 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 . . Rtrn RETURN	;1 instruction, 4 cycles

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 are 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 #16 Optimizing Destinations**

- Destination bit determines W for F for result
- Look at data movement and restructure

**Example 16-1**

Example: $A + B \rightarrow A$			
MOVWF	A, W	MOVWF	B, W
ADDWF	B, W	ADDWF	A, F
MOVWF	A		
3 instructions		2 instructions	

Careful use of the destination bits in instructions can save program memory. Here, register A and register B are summed and the result is put into the A register. A destination option is available for logic and arithmetic operations. In the first example, the result of the ADDWF instruction is placed in the working register. A MOVWF instruction is used to move the result from the working register to register A. In the second example, the ADDWF instruction uses the destination bit to place the result into the A register, saving an instruction.

**TIP #17 Conditional Bit Set/Clear**

- To move single bit of data from REGA to REGB
- Precondition REGB bit
- Test REGA bit and fix REGB if necessary

**Example 17-1**

BTFSS	REGA, 2	BCF	REGB, 5
BCF	REGB, 5	BTFSC	REGA, 2
BTFSC	REGA, 2	BSF	REGB, 5
BSF	REGB, 5		
4 instructions		3 instructions	

One technique for moving one bit from the REGA register to REGB is to perform bit tests. In the first example, the bit in REGA is tested using a BTFSS instruction. If the bit is clear, the BCF instruction is executed and clears the REGB bit, and if the bit is set, the instruction is skipped. The second bit test determines if the bit is set, and if so, will execute the BSF and set the REGB bit, otherwise the instruction is skipped. This sequence requires four instructions.

A more efficient technique is to assume the bit in REGA is clear, and clear the REGB bit, and test if the REGA bit is clear. If so, the assumption was correct and the BSF instruction is skipped, otherwise the REGB bit is set. The sequence in the second example uses three instructions because one bit test was not needed.

One important point is that the second example will create a two-cycle glitch if REGB is a port outputting a high. This is caused by the BCF and BTFSC instructions that will be executed regardless of the bit value in REGA.

## TIP #18 Swap File Register with W

### Example 18-1

```

SWAPWF    MACRO    REG
           XORWF   REG, F
           XORWF   REG, W
           XORWF   REG, F
           ENDM
    
```

The following macro swaps the contents of W and REG without using a second register.

Needs: 0 TEMP registers  
 3 Instructions  
 3 TCY

An efficient way of swapping the contents of a register with the working register is to use three XORWF instructions. It requires no temporary registers and three instructions. Here’s an example:

W	REG	Instruction
10101100	01011100	XORWF REG,F
10101100	11110000	XORWF REG,W
01011100	11110000	XORWF REG,F
01011100	10101100	Result

## TIP #19 Bit Shifting Using Carry Bit

Rotate a byte through carry without using RAM variable for loop count:

- Easily adapted to serial interface transmit routines.
- Carry bit is cleared (except last cycle) and the cycle repeats until the zero bit sets indicating the end.

### Example 19-1

```

LIST P=PIC12f629
INCLUDE P12f629.INC
buffer    equ    0x20

bsf      STATUS,C    ;Set 'end of loop' flag
rlf      buffer,f    ;Place first bit into C
bcf      GPIO,Dout   ;precondition output
btfsc   STATUS,C     ;Check data 0 or 1 ?
bsf      GPIO,Dout   ;Set data 0 or 1 ?
bcf      STATUS,C    ;Clear data in C
rlf      buffer,f    ;Place next bit into C
movf    buffer,f     ;Force Z bit
btfss   STATUS,Z     ;Exit?
goto    Send_Loop
    
```

# CHAPTER 2

## PIC<sup>®</sup> 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<sup>®</sup> 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.

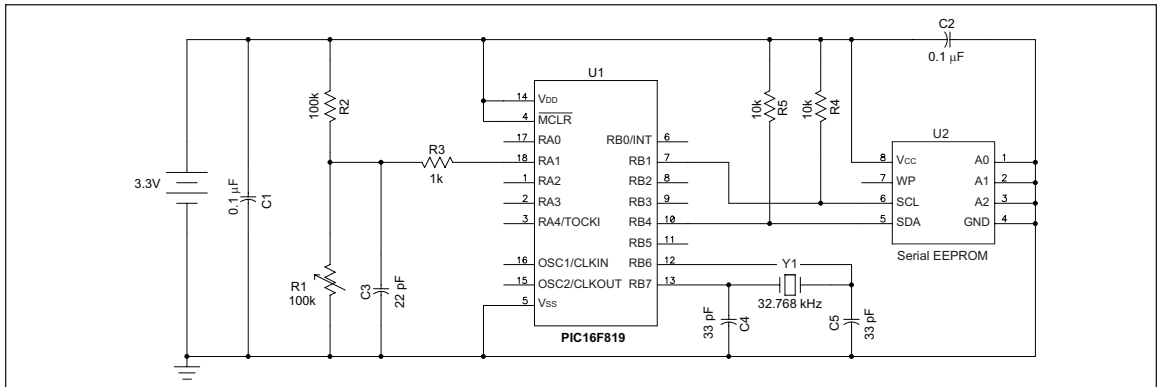
### GENERAL LOW POWER TIPS ‘N TRICKS

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

### TIP #1 Switching Off External Circuits/Duty Cycle

All the low power modes in the world won't help your application if you are unable to control the power used by circuits external to the microprocessor. Lighting an LED is equivalent to running most PIC MCUs at 5V-20 MHz. When you are designing your circuitry, decide what physical modes or states are required and partition the electronics to shutdown unneeded circuitry.

Figure: 1-1

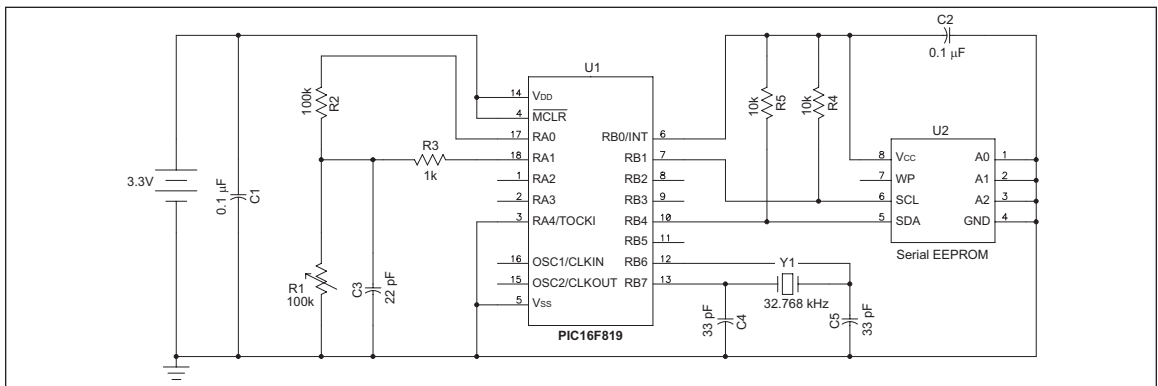


### Example:

The application is a long duration data recorder. It has a sensor, an EEPROM, a battery and a microprocessor. Every two seconds, it must take a sensor reading, scale the sensor data, store the scaled data in EEPROM and wait for the next sensor reading.

The system shown above is very simple and clearly has all the parts identified in the requirements. Unfortunately, it has a few problems in that the EEPROM, the sensor, and its bias circuit, are energized all the time. To get the minimum current draw for this design, it would be advantageous to shutdown these circuits when they are not required.

Figure: 1-2



In Figure 1-2, I/O pins are used to power the EEPROM and the sensor. Many PIC MCU devices can source up to 20 mA of current from each I/O, so there is no need to provide additional components to switch the power.

If more current than can be sourced by the PIC MCU is required, the PIC MCU can instead enable and disable a MOSFET to power the circuit. Refer to the data sheet for drive capabilities for a specific device.

## 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 Sensor Off EEPROM Off	1989	0.00005 0 0	5.00E-05	9.95E-05
Initialize MCU Sleep Sensor On EEPROM Off	1	0.00005 0.0165 0	1.66E-02	1.66E-05
Sample Sensor MCU Run Sensor On EEPROM Off	1	0.048 0.0165 0	6.45E-02	6.45E-05
Scaling MCU Run Sensor Off EEPROM Off	1	0.048 0 0	4.80E-02	4.80E-05
Storing MCU Run Sensor Off EEPROM On	8	0.048 0 1	1.05E+00	8.38E-03
<b>Total</b>	<b>2000</b>	<b>—</b>	<b>—</b>	<b>8.61E-03</b>

Average Current

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

$$= 0.0043 \text{ mA}$$

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 #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 k $\Omega$ ) 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<sup>2</sup>C™ signals, switches and for resistor dividers. For example, a typical I<sup>2</sup>C pull-up is 4.7k. However, when the I<sup>2</sup>C is transmitting and pulling a line low, this consumes nearly 700  $\mu$ A of current for each bus at 3.3V. By increasing the size of the I<sup>2</sup>C pull-ups to 10k, this current can be halved. The tradeoff is a lower maximum I<sup>2</sup>C 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.

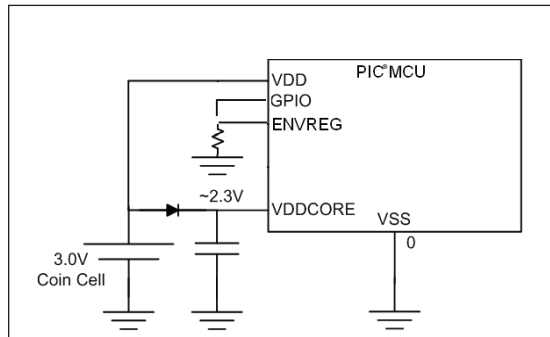
## TIP #6 Use an External Source for CPU Core Voltage

Some PIC MCUs such as “J” type devices (ex. PIC18F87J90 or PIC24FJ64GA004) use separate power for CPU core. These devices have an internal voltage regulator that can be used to provide the core voltage. Alternatively, the core voltage can be provided externally by disabling the internal regulator. In some cases, it is more power efficient to use an external source for the core. This is because the internal regulator powers the core at the nominal voltage that allows full speed operation. However, if an application doesn't require full speed, it is beneficial to use lower voltage to power the core. Disabling the internal regulator also turns off the BOR and LVD circuits, which saves power as well. The following examples show two different battery powered applications where it can be beneficial to disable the internal regulator.

### Example 1: Constant Voltage Source

When using a regulated power source or a battery with a flat discharge curve, such as a lithium coin cell, the regulator can be disabled and the core powered directly from the battery through a diode. The diode provides the voltage drop necessary to power the core at the correct voltage. It may be necessary to use a zener diode with a higher forward voltage for applications using sleep mode, as the current consumed in sleep is too low to cause the full forward voltage drop which can result in applying a voltage too high for the core.

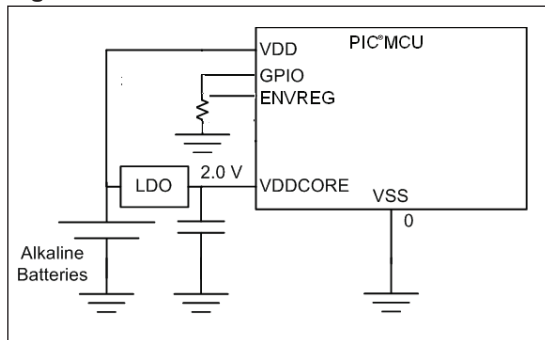
Figure 6-1:



### Example 2: Non-Constant Voltage Source

If the source for V<sub>DD</sub> is not constant, a regulator will be required. It can be beneficial to use an external low quiescent current regulator, which can be selected to provide lower voltage to the core than the internal regulator. Additionally, devices such as the MCP1700, which consumes 1 uA quiescent current while asleep, require less power than the internal regulator.

Figure 6-2:

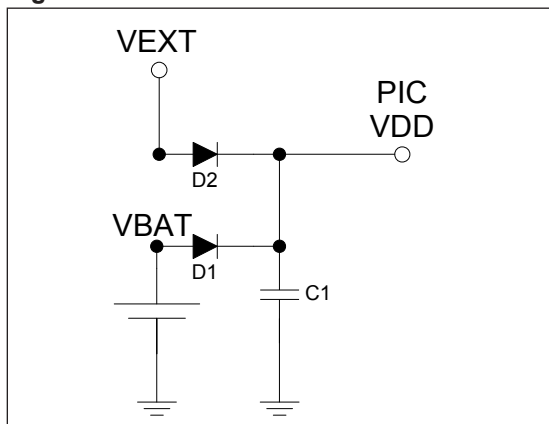




## TIP #7 Battery Backup for PIC MCUs

For an application that can operate from either an external supply or a battery backup, it is necessary to be able to switch from one to the other without user intervention. This can be accomplished with battery backup ICs, but it is also possible to implement with a simple diode OR circuit, shown in Figure 7-1. Diode D1 prevents current from flowing into the battery from VEXT when the external power is supplied. D2 prevents current from flowing into any external components from the battery if VEXT is removed. As long as the external source is present and higher voltage than the battery, no current from the battery will be used. When VEXT is removed and the voltage drops below VBAT, the battery will start powering the MCU. Low forward voltage Schottky diodes can be used in order to minimize the voltage dropout from the diodes. Additionally, inputs can be referenced to VEXT and VBAT in order to monitor the voltage levels of the battery and the external supply. This allows the micro to enter lower power modes when the supply is removed or the battery is running low. In order to avoid glitches on V<sub>DD</sub> caused by the diode turn-on delay when switching supplies, ensure enough decoupling capacitance is used on V<sub>DD</sub> (C1).

Figure 7-1:



## Dynamic Operation Tips n' Tricks

The following tips and tricks apply to methods of improving the dynamic operating current consumption of an application. This allows an application to get processing done quicker which enables it to sleep more and will help reduce the current consumed while processing.

## TIP #8 Enhanced PIC16 Mid-Range Core

The Enhanced PIC16 mid-range core has a few features to assist in low power. New instructions allow many applications to execute in less time. This allows the application to spend more time asleep and less time processing and can provide considerable power savings. It is important not to overlook these new instructions when designing with devices that contain the new core. The Timer1 oscillator and WDT have also been improved, now meeting nanoWatt XLP requirements and drawing much less current than in previous devices.

## TIP #9 Two-Speed Start-Up

Two-speed startup is a useful feature on some nanoWatt and all nanoWatt XLP devices which helps reduce power consumption by allowing the device to wake up and return to sleep faster. Using the internal oscillator, the user can execute code while waiting for the Oscillator Start-up (OST) timer to expire (LP, XT or HS modes). This feature (called “Two-Speed Start-up”) is enabled using the IESO configuration bit. A Two-Speed Start-up will clock the device from an internal RC oscillator until the OST has expired. Switching to a faster internal oscillator frequency during start-up is also possible using the OSCCON register. The example below shows several stages on how this can be achieved. The number of frequency changes is dependent upon the designer’s discretion. Assume a 20 MHz crystal (HS Mode) in the PIC16F example below.

### Example:

<u>T<sub>CY</sub></u> (Instruction Time)	<u>Instruction</u>	
	ORG 0x05	;Reset vector
125 μs @ 32 kHz	BSF STATUS,RP0	;bank1
125 μs @ 32 kHz	BSF OSCCON,IRCF2	;switch to 1 MHz
4 μs @ 1 MHz	BSF OSCCON,IRCF1	;switch to 4 MHz
1 μs @ 4 MHz	BSF OSCCON,IRCF0	;switch to 8 MHz
500 ns	application code	
500 ns	application code	
...	....	
..	...	
(eventually OST expires, 20 MHz crystal clocks the device)		
200 ns	application code	
...	....	
..	...	

## TIP #10 Clock Switching

Some nanoWatt devices and all nanoWatt XLP devices have multiple internal and external clock sources, as well as logic to allow switching between the available clock sources as the main system clock. This allows for significant power savings by choosing different clocks for different portions of code. For example, an application can use the slower internal oscillator when executing non-critical code and then switch to a fast high-accuracy oscillator for time or frequency sensitive code. Clock switching allows much more flexible applications than being stuck with a single clock source. Clock switching sequences vary by device family, so refer to device data sheets or Family Reference Manuals for the specific clock switching sequences.

## TIP #11 Use Internal RC Oscillators

If frequency precision better than ±5% is not required, it is best to utilize the internal RC oscillators inside all nanoWatt and nanoWatt XLP devices. The internal RC oscillators have better frequency stability than external RC oscillators, and consume less power than external crystal oscillators. Additionally, the internal clock can be configured for many frequency ranges using the internal PLL module to increase frequency and the postscaler to reduce it. All these options can be configured in firmware.