MICROCHIP

Tips 'n Tricks

8-pin FLASH
PIC® Microcontrollers
Outperform the Competition

Table of Contents

Tips ‘n Tricks With Hardware

Tips 'n Tricks With Software

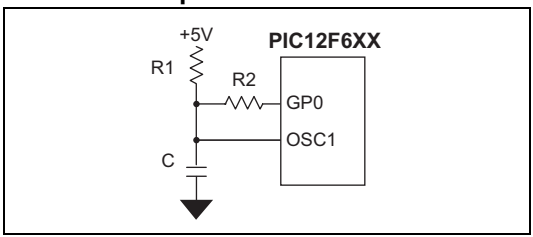
TIPS 'N TRICKS WITH HARDWARE

Microchip continues to provide innovative products that are smaller, faster, easier to use and more reliable. The 8-pin FLASH PICmicro® 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 PICmicro 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.

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



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

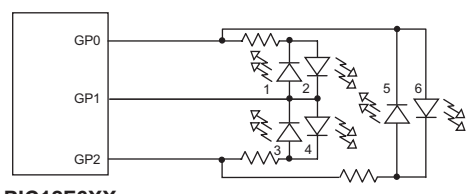
TIP #2 Input/Output Multiplexing

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 x (GP - 1).

Example – Six LEDs on three I/O pins

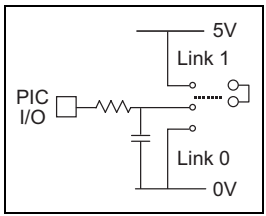
GPx			LEDs					
0	1	2	1	2	3	4	5	6
0	0	0	0	0	0	0	0	0
0	1	Z	1	0	0	0	0	0
1	0	Z	0	1	0	0	0	0
Z	0	1	0	0	1	0	0	0
Z	1	0	0	0	0	1	0	0
0	Z	1	0	0	0	0	1	0
1	Z	0	0	0	0	0	0	1
0	0	1	0	0	1	0	1	0
0	1	0	1	0	0	1	0	0
0	1	1	1	0	0	0	1	0
1	0	0	0	1	0	0	0	1
1	0	1	0	1	1	0	0	0
1	1	0	0	0	0	1	0	1
1	1	1	0	0	0	0	0	0



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

State	Link0	Link1
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.

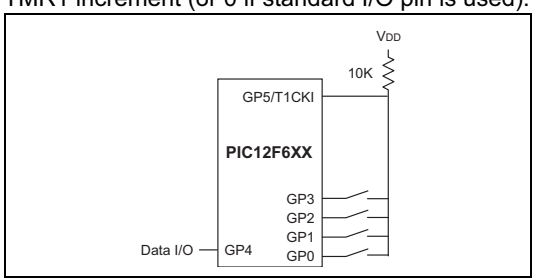
TIP #4 Reading DIP Switches

The input of a timer can be used to test which switch(s) is closed. The input of Timer 1 is held high with a pull-up resistor. Sequentially, each switch I/O is set to input and Timer 1 is checked for an increment indicating the switch is closed.

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

Each bit in the DP register represents its corresponding switch position. By setting Timer 1 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).



TIP #5 Scanning Many Keys With One Input

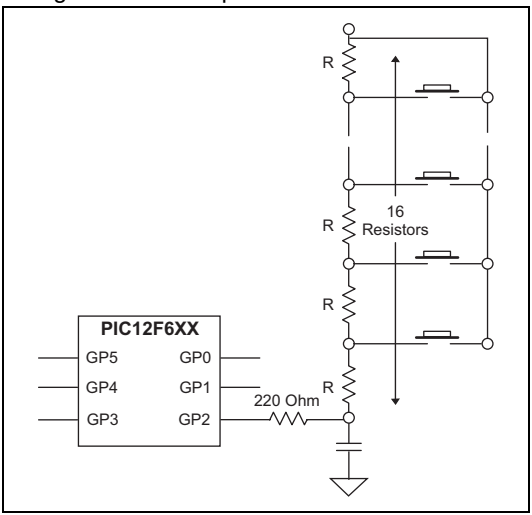
The time required to charge a capacitor depends on resistance between V_{DD} and capacitor. When a button is pressed, V_{DD} is supplied to a different point in the resistor ladder. The resistance between V_{DD} and the capacitor is reduced, which reduces the charge time of the capacitor. A timer is 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.

TIP #5 Scanning Many Keys With One Input (Cont.)

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



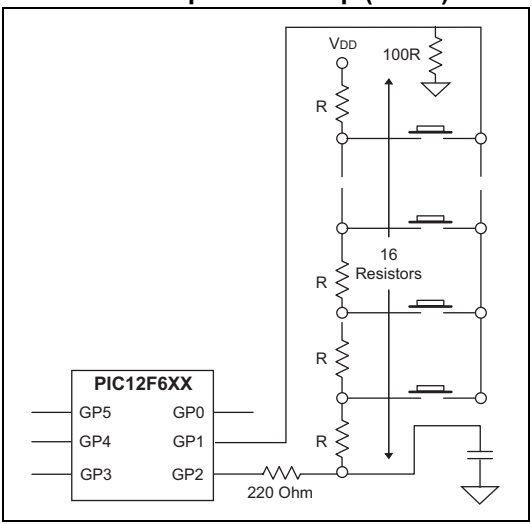
See AN512 *Implementing Ohmmeter/ Temperature Sensor* for code ideas.

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 Ohm resistor. GP1 is set to output high and GP2 is set to an input to measure the capacitor charge time.

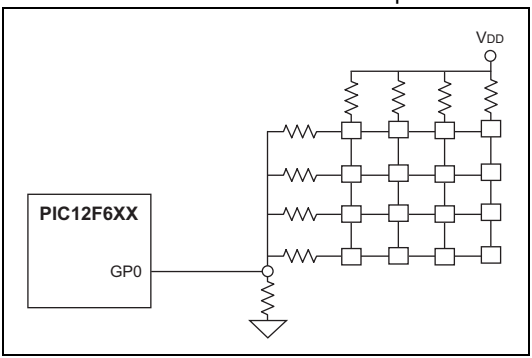
- GP1 pin connected to key common
- Enable wake-up on port change
- Set GP1 as input and GP2 high prior to sleep
- If key is pressed the PIC® MCU wakes up, GP2 must be set low to discharge capacitor
- Set GP1 high upon wake-up to scan keystroke

TIP #6 Scanning Many Keys and Wake-up From Sleep (Cont.)



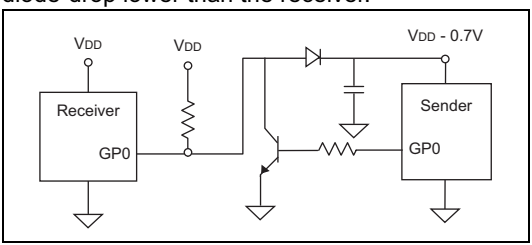
TIP #7 4x4 Keyboard with 1 Input

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.



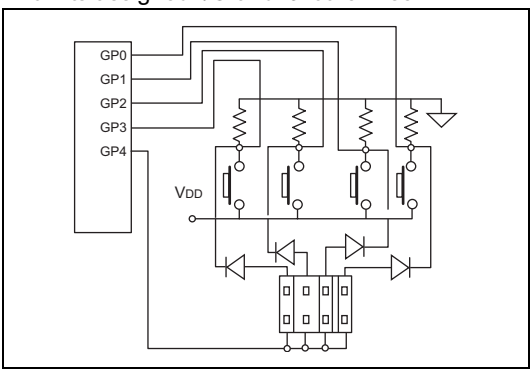
TIP #8 One Pin Power/Data

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 VDD. 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. VDD is supplied to the sender through the data line. The capacitor stabilizes the sender's VDD and a diode prevents the capacitor from discharging through the I/O line while it is low. Note that the VDD of the sender is a diode-drop lower than the receiver.



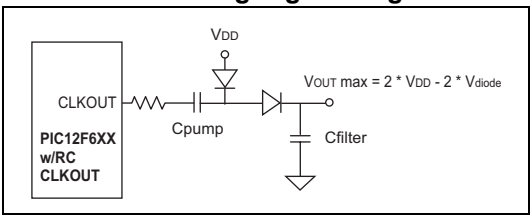
TIP #9 Decode Keys and ID Settings

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.



- When GP4 = 1 and no keys are pressed, read ID setting
- When GP4 = 0, read the switch buttons

TIP #10 Generating High Voltages



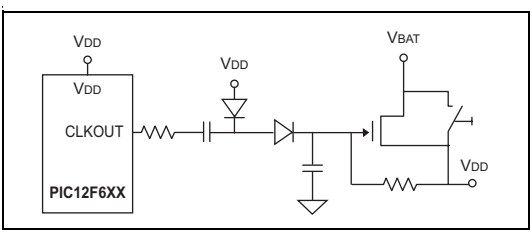
Voltages greater than VDD can be generated using a toggling I/O. PIC® MCUs CLKOUT/OSC2 pin toggles at one quarter the frequency of OSC1 when in external RC oscillator mode. When OSC2 is low, the VDD diode is forward biased and conducts current, thereby charging Cpump. After OSC2 is high, the other diode is forward biased, moving the charge to Cfilter. The result is a charge equal to twice the VDD minus two diode drops. This can be used with a PWM, a toggling I/O or other toggling pin.

TIP #11 V_{DD} Self Starting Circuit

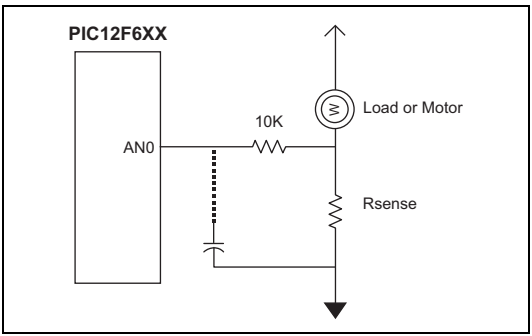
Building on the previous topic, the same charge pump can be used by the MCU to supply its own V_{DD}. Before the switch is pressed, V_{BAT} has power and the V_{DD} points are connected together but unpowered. When the button is pressed, power is supplied to V_{DD} and the MCUs CLKOUT (in external RC oscillator mode) begins toggle. The voltage generated by the charge pump turns on the FET allowing V_{DD} 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



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



- 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 (Rsense) 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

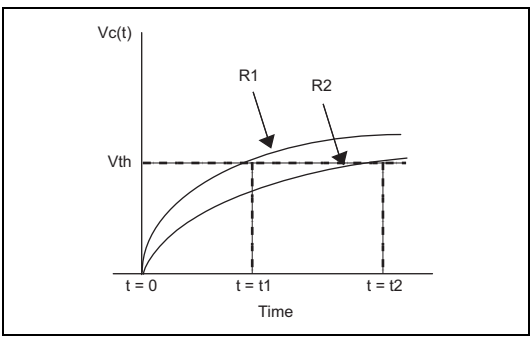
1. RC timing method with reference resistor
2. Charge balancing method
3. A/D method

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. These next topics will cover ways of dealing with these factors for getting the most out of a PIC® MCU.

TIP #13 Reading A Sensor With Higher Accuracy – RC Timing Method

RC Timing Method:

Simple RC step response
$Vc(t) = V_{DD} * (1 - e^{-t/(RC)})$
$t = -RC \ln(1 - Vth/V_{DD})$
Vth/V_{DD} is constant
$R2 = (t2/t1) * R1$



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.

TIP #13 Reading A Sensor With Higher Accuracy – RC Timing Method (Cont.)

Application Notes:
AN512 *Implementing Ohmmeter/Temperature Sensor*
AN611 *Resistance and Capacitance Meter Using a PIC16C622*

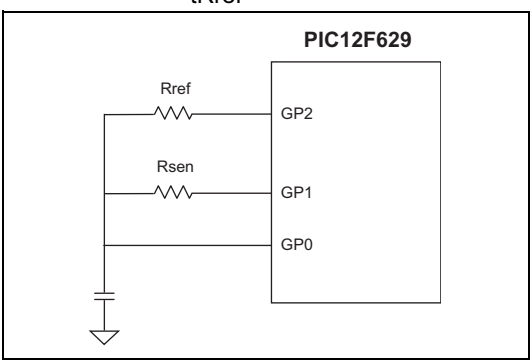
Here is the schematic and software flow for using a reference resistor to improve the accuracy of an analog sensor reading. The reference resistor (Rref) and sensor (Rsen) are assigned an I/O and share a common capacitor. GP0 is used to discharge the capacitor and represents the capacitor voltage.

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.

TIP #13 Reading A Sensor With Higher Accuracy – RC Timing Method (Cont.)

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 tRsen (GP0 changes to 1)
4. Repeat step 1
5. Set GP0 to an input and GP2 to a high output
6. Measure tRref (GP0 changes to 1)
7. Use film polypropylene capacitor
8. Rth = x Rref $\frac{tRsen}{tRref}$



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

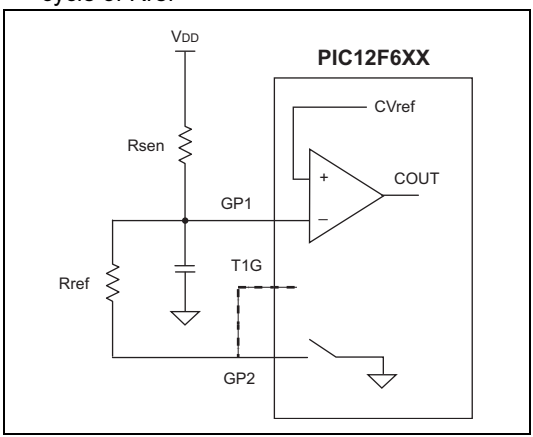
TIP #13 Reading A Sensor With Higher Accuracy – Charge Balancing Method

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 use. 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 (CVref) on the PIC12F629/675 are ideal for this application.

TIP #13 Reading A Sensor With Higher Accuracy – Charge Balancing Method (Cont.)

1. GP1 average voltage = CVref
2. Time base as sampling rate
3. At the end of each time base period:
   - If GP1 > CVref, then GP2 Output Low
   - If GP1 < CVref, 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 Rref



TIP #13 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.

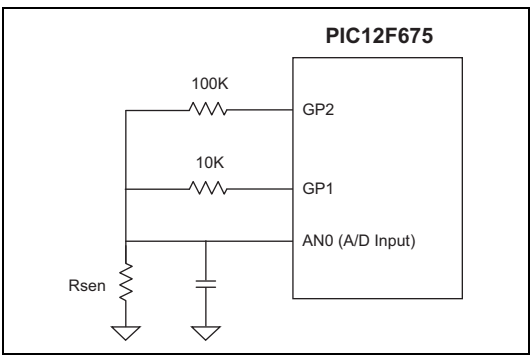
Summary:

High range: GP1 output '1' and GP2 input

Low range: GP1 input and GP2 output '1'

TIP #13 Reading A Sensor With Higher Accuracy – A/D Method (Cont.)

1. 10K and 100K resistors are used to set the range
2. Vref for A/D = V_{DD}
3. Rth calculation is independent of V_{DD}
4. Count = Rsen/(Rsen+Rref) x 255
5. Don't forget to allow acquisition time for the A/D

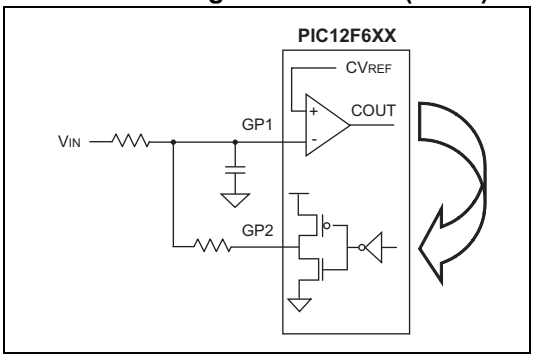


TIP #14 Delta Sigma Converter

The charge on the capacitor on GP1 is maintained about equal to the CVref by the MCU monitoring COUT and switching GP2 from input mode or output low appropriately. A timer is used to sample the COUT 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.

TIP #14 Delta Sigma Converter (Cont.)



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

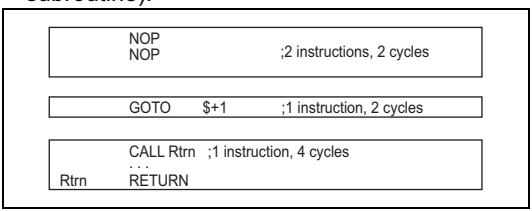
NOTES:

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).



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

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

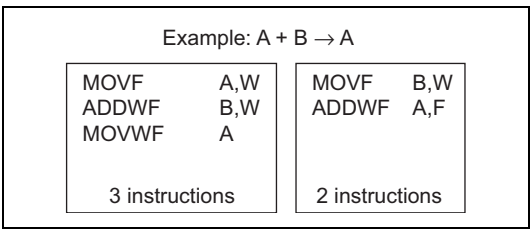
TIP #15 Delay Techniques (Cont.)

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. 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

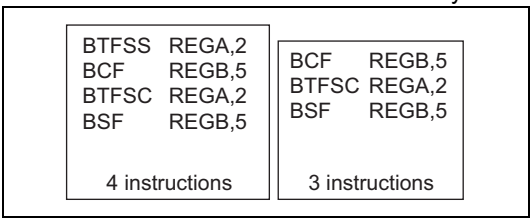
- Destination bit determines W for F for result
- Look at data movement and restructure



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



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.

TIP #17 Conditional Bit Set/Clear (Cont.)

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

```
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.

```
                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
        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
```

NOTES:

NOTES:

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

The graphics in this document are for illustration only. Microchip reserves the right to modify the contents of its development systems.

Trademarks

The Microchip name and logo, the Microchip logo, KEELOQ, MPLAB, PIC, PICmicro, PICSTART and PRO MATE are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, microID, MXDEV, MXLAB, PICMASTER, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

dsPIC, dsPICDEM.net, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, PICC, PICDEM, PICDEM.net, PICkit, rfPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

Serialized Quick Turn Programming (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.

Printed on recycled paper.

Worldwide Sales and Service

AMERICAS

Corporate Office
Tel: 480-792-7200
Technical Support:
480-792-7627

Rocky Mountain
Tel: 480-792-7966

Atlanta
Tel: 770-640-0034

Boston
Tel: 978-692-3848

Chicago
Tel: 630-285-0071

Dallas
Tel: 972-818-7423

Detroit
Tel: 248-538-2250

Kokomo
Tel: 765-864-8360

Los Angeles
Tel: 949-263-1888

San Jose
Tel: 408-436-7950

Toronto
Tel: 905-673-0699

ASIA/PACIFIC

Australia
Tel: 61-2-9868-6733

China-Beijing
Tel:86-10-85282100

China-Chengdu
Tel: 86-28-86766200

China-Fuzhou
Tel: 86-591-7503506

China-Hong Kong SAR
Tel: 852-2401-1200

China-Shanghai
Tel: 86-21-6275-5700

China-Shenzhen
Tel: 86-755-82901380

China-Qingdao
Tel: 86-532-5027355

India
Tel: 91-80-2290061

Japan
Tel: 81-45-471- 6166

Korea
Tel: 82-2-554-7200

Singapore
Tel: 65-6334-8870

Taiwan
Tel: 886-2-2717-7175

EUROPE

Austria
Tel: 43-7242-2244-399

Denmark
Tel: 45-4420-9895

France
Tel: 33-1-69-53-63-20

Germany
Tel: 49-89-627-144-0

Italy
Tel: 39-039-65791-1

United Kingdom
Tel: 44-118-921-5869

12/05/02

DNV Certification, Inc. USA — ANSI-RAB QMS — ACCREDITED — DNV — DNV MSC The Netherlands Accredited by the RvA — ISO 9001 / QS-9000 REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, non-volatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

Microchip Technology Inc.
2355 W. Chandler Blvd. • Chandler, AZ 85224 U.S.A.
Phone: 480-792-7200 • Fax: 480-792-9210
www.microchip.com

 DS40040B