AN1202

Capacitive Sensing with PIC10F

Author: *Marcel Flipse*
Microchip Technology Inc.

INTRODUCTION

This application note describes a method of implementing capacitive sensing on the PIC10F204/6 family of controllers. It assumes general knowledge of the sensing process; it is also recommended that application note AN1101*, "Introduction to Capacitive Sensing"*, be read in order to understand the hardware concepts.

PIC10F204 and PIC10F206 microcontrollers have an onboard comparator that can be used for capacitive sensing of a single key.

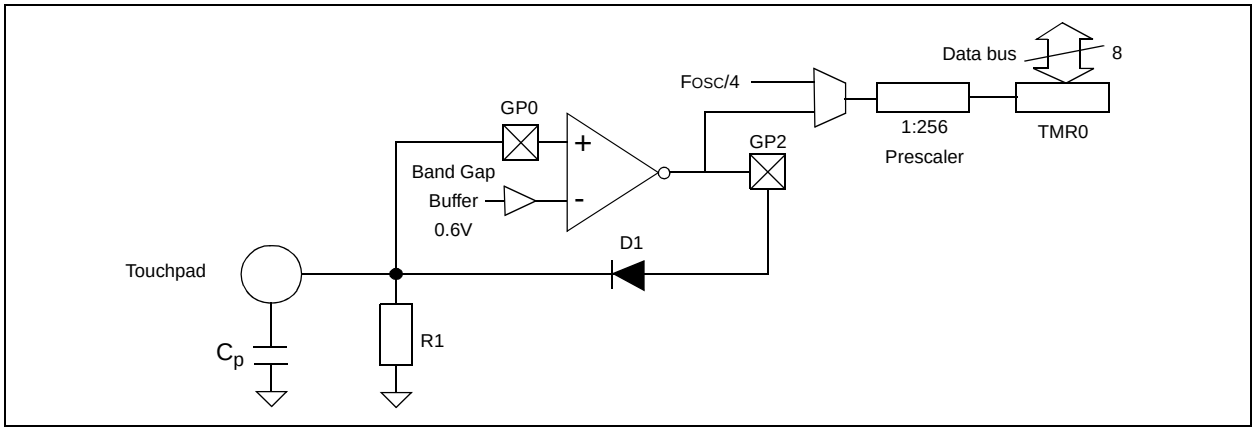
IMPLEMENTATION

Capacitive sensing is implemented by turning the comparator into a relaxation oscillator. The output of the comparator is used to charge and discharge the sensing capacitor, that is formed by a pad on the circuit board. The charge rate is determined by the RC time constant, created by an external resistor and the capacitance of the pad.

Introduction of additional capacitance from a person's finger to ground causes a frequency change. This change is measured by the PIC® MCU and processed to detect a finger press.

The basic oscillator circuit is shown in Figure 1. C_p is the parasitic capacitance. During start-up this capacitance has no charge and the voltage is zero. Therefore, the output of the comparator will be high and the touch pad is rapidly charged through D1 until it reaches V_{DD}.

FIGURE 1: BASIC OSCILLATOR SCHEMATIC

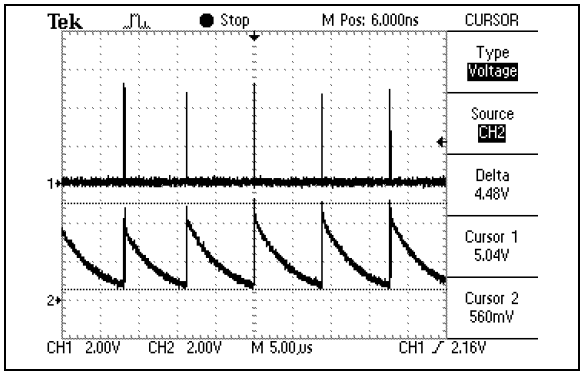


The output of the comparator will change to the low state. Then, it discharges slowly through R1 until it reaches the trip point of the internal band gap reference of 0.6V. The output of the comparator will go high again and the cycle repeats itself.

A scope plot of this charge/discharge cycle can be seen in Figure 2. Trace 1 shows the output of the comparator and trace 2 the voltage across the pad (C_p). The full circuit schematic is illustrated in Appendix A.

The output of the comparator is a frequency that is related to the capacitance of the pad. A base frequency of 350 kHz is used in this example. Any frequency in the 100-400 kHz range will work. Using a higher frequency makes the measurement cycle shorter.

FIGURE 2: OSCILLATOR OUTPUT



MEASURING FREQUENCY

Once the oscillator is constructed, its frequency must be monitored to detect a drop in frequency caused by a finger press. To measure the frequency, the oscillator is started and the output of the comparator fed into TMR0. TMR0 is an 8-bit timer/counter with an 8-bit software programmable prescaler. After a fixed software delay, the prescaler and the value of TMR0 are read. Reading both the prescaler and the TMR0 value will give you a 16-bit value of the frequency of the oscillator (frequency in counts).

In order to read the prescaler directly for a PIC10F, a software technique is used to estimate the value of the prescaler. After the measurement, the relaxation oscillator is stopped and the clock source for TMR0 is set to the internal oscillator (FOSC/4). The software then polls for a increase or roll-over of the TMR0 value. The amount of time it takes for TRM0 to change value is an indication of the prescaler value.

Thus, the following sequence is needed to measure the frequency:

1. Turn on the oscillator
2. Clear TMR0 and the prescaler
3. Wait a fixed time duration (100 ms in Example 2)
4. Stop the oscillator
5. Read the TMR0 value
6. Select FOSC/4 as the clock source for TMR0
7. Count the number of cycles it takes before TMR0 changes value, to get an estimate of the prescaler

SOFTWARE

The detection scheme used to detect a finger press is based on the principle that there is rapid drop in frequency counts from the running average. If a finger touches the pad, the capacitance increases and the frequency drops.

To initialize the oscillator, the following sequence is needed:

EXAMPLE 1: INITIALIZATION CODE

```
MOVLW  b'11111001' ;set gp1,gp2 as an output
TRIS   gpio
MOVLW  b'11110111'
      ;  ||||||||_ ps0
      ;  |||||||__ ps1
      ;  ||||||___ ps2set prescaler to 1:256
      ;  |||||____ psaprescaler assigned to tmr0
       ; ||||_____ t0se  increment on high to low
       ; |||______ t0cs  transition on t0cki
       ; ||_______ #gppu pull-ups disabled
       ; |________ #gpwu wake-up pin change disabled
OPTION

MOVLW  b'00001011'
       ; ||||||||_ #cwu wake-up on comp ch. disabled
       ; |||||||__ cpref   pos ref is cin+
       ; ||||||___ cnref   neg ref is internal 0.6V
       ; |||||____ cmpon   comparator on
       ; ||||_____ cmpt0cs comp. used as tmr0 source
       ; |||______ pol     output is inverted
       ; ||_______ #couten output is placed on cout
       ; |________ cmpout  -read only bit-
MOVWF  cmcon0
CLRF   tmr0       ; clear tmr0 and the 1:256 prescaler
```

After this sequence, the oscillator is turned on and the prescaler and TMR0 will increment. Longer or shorter discharge times can be obtained by varying the value of R1.

In this example, the software waits 100 ms and stops the oscillator. The 100 ms was chosen to obtain a large value in the prescaler and TMR0. Choosing a different base frequency for the oscillator may require a different delay.

Make sure the delay is chosen long enough to get a good reading, but short enough so that TMR0 does not overflow.

EXAMPLE 2:

```
MOVLW gatedtime   ;  constant equals 100
CALL  delay       ;  wait 100 mSec

BCF   cmcon0,cmpon; turn off oscillator

MOVF  tmr0,w      ;  high byte of freq value
                  ;  is stored in tmr0
MOVWF freqhi      ;  low value is still in
                  ;  the prescaler
```

The value of the prescaler is not directly readable. To get an estimate of the prescaler, the clock source for TMR0 is changed to FOSC/4 and a software loop counts the time needed for TMR0 to increment or roll over.

EXAMPLE 3:

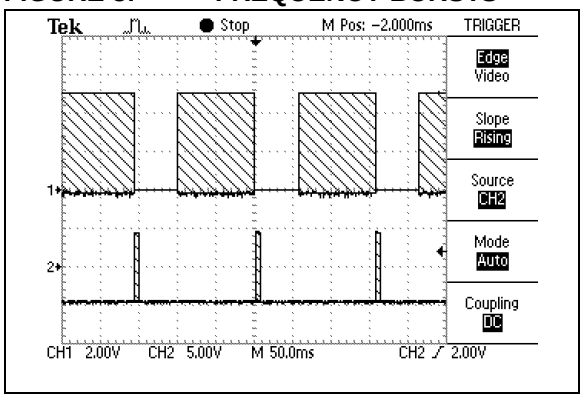
```
   MOVLW   b'11010111'; change clock
                      ; source to Fosc/4
   OPTION

measureprescaler:
   INCF    freqlo    ; was initialised to 255 and
                     ; set to 0 here
   MOVF    tmr0,w    ; get the current value of tmr0
   XORWF   freqhi,w  ; compare it with the original
                     ; value of tmr0
   BTFSC   status,z  ; did tmr0 increment?
   GOTO    measureprescaler; no, loop and increment
```

This loop takes 6 instruction cycles, so the maximum value for `freqlo` will be 43. This value is multiplied by 6 and clipped to 255. The two Least Significant bits (LSb) are not useful and, therefore, the result is divided by 4.

Figure 3 is a snapshot of the free running oscillator. The upper trace shows the oscillator being turned on periodically for 100 ms. The lower trace shows the PIC microcontroller transmitting the real time data serially over the free available pin.

FIGURE 3: FREQUENCY BURSTS



DETECTING A FINGER PRESS

At this point the system is complete, except for the detection and signaling of a button press. The remaining portion is handled in the main loop of the program.

A simple way to watch for the decrease in frequency is to use two variables and a constant. These are:

EXAMPLE 4:

```
freqhi:freqlo           ; var Current sensor data
averagehi:averagelo     ; var Running Average
triphi:triplo           ; const Trip point
```

`freqhi:freqlo` holds the current sensor data.

`averagehi:averagelo` is the running average of previous samples, calculated as follows:

EQUATION 1:

$$\frac{((2^n) - 1)\ \text{x averagevalue} + \text{currentvalue}}{2^n}$$

For example, if n is set to 4, the current reading is given a weight of $1/16^{th}$, while the running average is weighed as $15/16^{th}$. It is not necessary to store 16 variables to do a 16-point average.

Using a number which is a power of 2 for the N-point average saves processing time because right-shifts can be used instead of software division.

The simplest button press algorithm would be to test if the current value is a fixed distance below the average as in the pseudocode example below.

EXAMPLE 5:

```
If (freq < (average - trip) then
        ; button is pressed
        ; user code here
Else
        ; button is not pressed
        ; user code here
EndIf
```

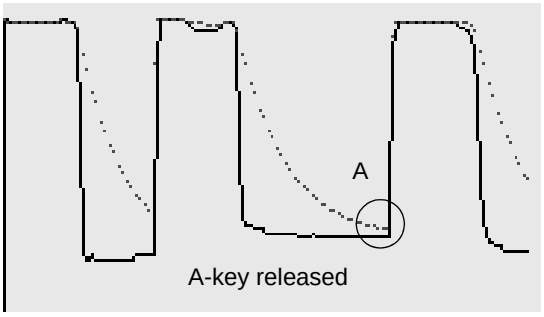
To provide an illustrative example, assume the oscillator reads 10,000 without a finger pressing the button. The average and current value will both be 10,000. As the designer, assume a trip value of 1,000 is a good value. When someone presses the button, the raw value immediately drops to 8,500, but the average was still at 10,000. The “if statement” in Example 5 will prove to be true, because 8,500 is less than 9,000. The button is pressed. Then, a flag may be set or a response performed in reaction.

Note:	The example above is very simplistic to demonstrate the frequency drop as the fundamental change common to all. Alternative software algorithms for detecting button presses can be found in the application note AN1103, “*Software Handling for Capacitive Sensing*”.

IMPLEMENTING CONTINUOUS TOUCH

Due to the averaging mechanism in the software, a finger press will be deactivated when the average value reaches the current value again. The red dotted line in Figure 4 is the average value, the black line the raw value. As can be seen, the average value is slowly tracking the current value. If the difference between the current value and the average value is less than the trip point, the key will be released.

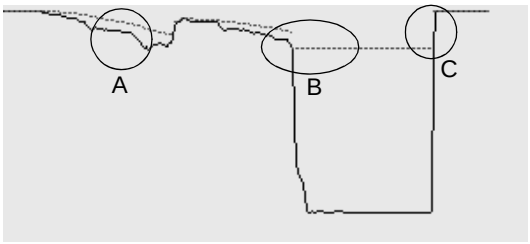
FIGURE 4: AVERAGING MECHANISM



To implement continuous touch, a different algorithm can be used. The averaging must cease to track the current value when it has crossed the trip threshold. To prevent a stuck key, an additional hysteresis is subtracted from the average value. Due to drift, the current value may not reach the same value as before the finger press. The average value locked after a finger press can be seen in Figure 5.

Slight changes will still be tracked.

FIGURE 5: CONTINUOUS TOUCH



A – Average value still tracking the current value

B – Keypress is detected and the average is locked and a constant value is subtracted from the average value

C – Key is released and the average algorithm is restarted

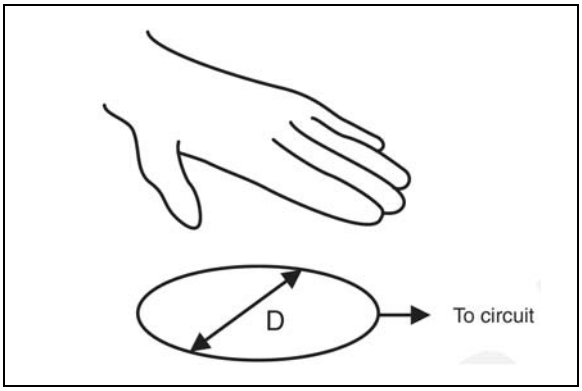
Refer to the firmware source code for more information on how to enable this feature.

IMPLEMENTING A PROXIMITY SWITCH

A proximity switch is a non-contact type switch. The typical use for a proximity switch is to sense the presence or absence of an object, like a hand, without actually contacting the object. This is useful for applications like electric hand dryers and door access control.

The circuit described can easily be turned into a proximity switch. This is done by using a larger pad as a sensing element and by adjusting the value of the discharging resistor, R1. The trip point (`triphi:triplo`) must also be adjusted it to make a proximity sensor. The trip point must be lowered significantly to make a proximity sensor instead of a touch sensor. As a rule of thumb, the maximum detectable distance from a hand to the sensor pad is equal to the diameter of the sensor pad. Thus, the larger the pad, the greater the distance. Any material in between the hand and the sensor may influence the maximum distance.

FIGURE 6: PROXIMITY SWITCH



The sensor can be a large copper area on a printed circuit board or can be constructed with conductive tape inside a plastic enclosure, therefore allowing a single or double curved surface. Even objects like a metal enclosure may be used as a sensor, as long as it is not physically connected to ground.

When using a large pad for the proximity switch, the capacitance will be larger than a standard button. Therefore, the frequency will be lower. Adjust the value of R1 so that the base frequency will remain within the 100 to 400 kHz range.

PRECAUTIONS

Timer0 Overflow

Since the principle measurement is read from the TRM0 value, TMR0 must not overflow. A longer period will allow more counts, but select a measurement period short enough that this does not happen. Increasing the oscillator frequency allows shorter measuring cycles without losing resolution.

Stuck Buttons

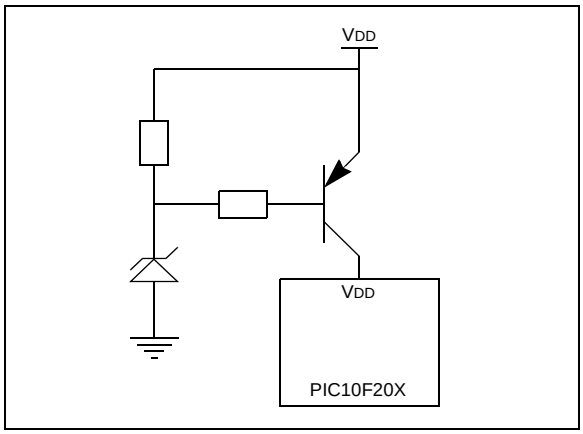
When implementing the continuous touch algorithm, the averaging mechanism will stop. Due to drift, the current value may not reach the same value. Make sure the hysteresis is large enough to compensate for the drift of the current value.

Power Supply Fluctuations

The trip point for the oscillator is the internal 0.6V reference. The capacitance is discharged from V_{DD} to 0.6V, therefore a rapid change in V_{DD} will cause the oscillator to change frequency. This could trigger false finger presses. Slow variations, like running of a battery, will be compensated by the averaging mechanism. If possible, use a regulated power supply and use decoupling capacitors close to the PIC microcontroller.

Also, take the V_{DD} rise time into account. If the minimum V_{DD} Rise Rate cannot be met, the device must be held in Reset until the operating parameters are met. Alternatively, a circuit shown in Figure 7 below can be used. This way, the $\overline{MCLR}$ pin can still be used as a general purpose input pin.

FIGURE 7: V_{DD} RISE TIME



CIRCUIT BOARD DESCRIPTION

The full schematic is illustrated in Appendix A. The board can be powered by an external power supply or by the serial port. The RTS (Request To Send) and DTR (Data Terminal Ready) pins can supply enough current to power the board. These pins are tied to an LDO through D3. The MCP1703 is used to make a stable 5V supply for the PIC MCU.

The free IO pin can be routed to a LED and buzzer, or it can be connected to the serial port by setting the jumper on the correct position of K3. A single transistor (Q1) is used to shift the voltage levels to an RS-232 compatible level. The negative level (V-) is derived from the PC's transmit pin, TX through D5.

J1 is a jumper that is used to switch between modes. With the jumper in place, the PIC10F transmits real time data, like the average value, the current value, the trip point and averaging depth. Without the jumper the circuit functions as a button and operates the LED and buzzer. Set jumper K3 to the correct position depending on the mode.

K7 is the programming connector. An ICD 2 or PICkit™ 2 can be used to program the board. Disconnect the programmer after programming. The PGD pin from the programmer is shared with the touch pad and inhibits correct operation of the free running oscillator.

CONCLUSIONS

Software is provided with this application note to aid in understanding and expediting design. The software to drive capacitive sensing can be either very simple or can handle complex algorithms for button detection.

Additional reference materials include:

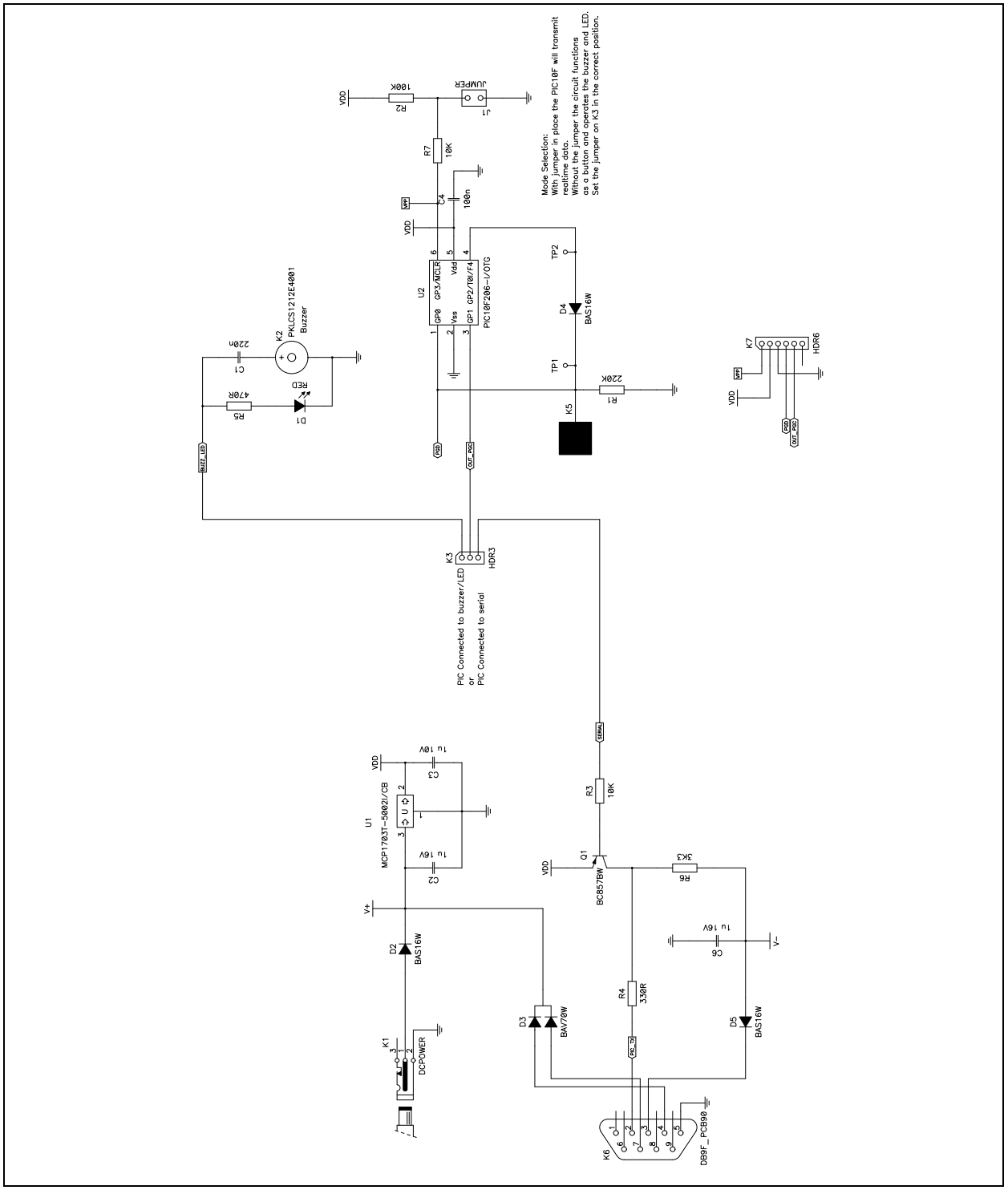
AN1101, "*Introduction to Capacitive Sensing*"

AN1102, "*Layout for Capacitive Sensing*"

AN1103, "*Software Handling for Capacitive Sensing*"

AN1104, "*Capacitive Mini-Button Configurations*"

Appendix A. Full Circuit Schematic



NOTES:

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.

Trademarks

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.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://support.microchip.com
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-572-9526
Fax: 886-3-572-6459

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/02/08