



mTouch™ Sensing Solution

User's 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in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates.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, PRO MATE, rPIC and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


FilterLab, 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, PICkit, PICDEM, PICDEM.net, PICtail, PIC³² logo, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, Select Mode, Total Endurance, UNI/O, 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.

© 2008, 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.

Table of Contents

Preface	1
Introduction	1
Document Layout.....	1
Conventions Used in this Guide	2
Recommended Reading	3
The Microchip Web Site.....	3
Customer Support.....	4
Document Revision History	4
 Chapter 1: Introduction	
What is mTouch™ Sensing Solution?	5
Inside This User's Guide.....	6
 Chapter 2: How mTouch™ Sensing Solution Works	
Basic Operation	7
 Chapter 3: Building an mTouch™ Sensing Solution System	
Development Tool Requirements	9
Getting Started	9
Existing Software Templates	9
Select A Device	9
Easily Creating Software	10
Software In Action.....	13
Developing Hardware	18
 Chapter 4: Software Details and Choices	
Overview.....	21
Initialization	21
ISR.....	21
Taking a Reading.....	21
Threshold Definitions	22
Decode	22
Averaging	22
Quick Release	22
Warm-Up	22
Setting the Next Sensor to Test.....	23

mTouch™ Sensing Solution User's Guide

Chapter 5: Reference Designs

Single Button Reference Design	25
Four Button Reference Design	25
Multi-Button Reference Design.....	25

Appendix A. Code Module Library

What is the Code Module Library?	27
How does the Code Module Library Work?	27
Introduction to Code Module Library	28
Code Module Library Items	29
Create A Snippet	30
Copy A Snippet.....	33
Edit A Snippet.....	34
Selecting Snippet Files Used.....	35
Tips and Tricks	37
Troubleshooting.....	38

Appendix B. Code Modules

Code Module List.....	39
Average – Always %.....	40
Average – Always 16.....	40
Average – Gated %	41
Average – Gated 16	41
CAPINIT 61x.....	42
CAPINIT 690 Family.....	42
CAPINIT 88X Family	43
Convert VAL to %	44
Convert VAL to 16	44
Decode – Paired Press.....	45
Decode – Press, Diff Thresh.	45
Decode – % Voting.....	46
Decode – Press %.....	46
Decode – Press 16.....	47
Decode – Press and Release.....	47
ISR Basic.....	48
ISR with I ² C™ Calls	48
Quick Release %	49
Quick Release 16	49
Reading – Get TMR1 Value	50
Restart Timers.....	50
SetNextChannel 4-Button Example.....	51

SetNextChannel 25-Button Example	51
Threshold – Absolute Counts	52
Threshold – Percentage	52
Warm-up %.....	53
Warm-up 16.....	53

mTouch™ Sensing Solution User's Guide

NOTES:

Preface

NOTICE TO CUSTOMERS

All documentation becomes dated, and this manual is no exception. Microchip tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our web site (www.microchip.com) to obtain the latest documentation available.

Documents are identified with a “DS” number. This number is located on the bottom of each page, in front of the page number. The numbering convention for the DS number is “DSXXXXA”, where “XXXX” is the document number and “A” is the revision level of the document.

For the most up-to-date information on development tools, see the MPLAB® IDE on-line help. Select the Help menu, and then Topics to open a list of available on-line help files.

INTRODUCTION

This chapter contains general information that will be useful to know before using the PIC18F1220/1320 (SDK). Items discussed in this chapter include:

- Document Layout
- Conventions Used in this Guide
- Recommended Reading
- The Microchip Web Site
- Customer Support
- Document Revision History

DOCUMENT LAYOUT

This document describes how to use the PIC18F1220/1320 as a development tool to emulate and debug firmware on a target board. The manual layout is as follows:

- **Chapter 1. Introduction**
- **Chapter 2. How mTouch™ Sensing Solution Works**
- **Chapter 3. Building an mTouch™ Sensing Solution System**
- **Chapter 4. Software Details and Choices**
- **Chapter 5. Reference Designs**
- **Appendix A. Code Module Library**
- **Appendix B. Code Modules**

mTouch™ Sensing Solution User's Guide

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic characters	Referenced books	<i>MPLAB® IDE User's Guide</i>
	Emphasized text	...is the <i>only</i> compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic text with right angle bracket	A menu path	<u><i>File>Save</i></u>
Bold characters	A dialog button	Click OK
	A tab	Click the Power tab
N'Rnnnn	A number in verilog format, where N is the total number of digits, R is the radix and n is a digit.	4'b0010, 2'hF1
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier New font:		
Plain Courier New	Sample source code	#define START
	Filenames	autoexec.bat
	File paths	c:\mcc18\h
	Keywords	_asm, _endasm, static
	Command-line options	-Opa+, -Opa-
	Bit values	0, 1
	Constants	0xFF, 'A'
Italic Courier New	A variable argument	<i>file.o</i> , where <i>file</i> can be any valid filename
Square brackets []	Optional arguments	mcc18 [options] <i>file</i> [options]
Curly brackets and pipe character: { }	Choice of mutually exclusive arguments; an OR selection	errorlevel {0 1}
Ellipses...	Replaces repeated text	var_name [, var_name...]
	Represents code supplied by user	void main (void) { ... }

RECOMMENDED READING

This user's guide describes how to use the mTouch™ Sensing Solution Software Developer's Kit (SDK). Other useful documents are listed below. The following Microchip documents are available and recommended as supplemental reference resources.

Readme Files

For the latest information on using other tools, read the tool-specific Readme files in the Readmes subdirectory of the MPLAB® IDE installation directory. The Readme files contain update information and known issues that may not be included in this user's guide.

Design Center

Microchip has a capacitive touch design center which can be found on **www.microchip.com/mtouch**.

The following Microchip Application Notes are available and recommended as supplemental reference resources.

AN1101, "*Introduction to Capacitive Sensing*," (DS01101)

AN1102, "*Layout and Physical Design Guidelines for Capacitive Sensing*," (DS01102)

AN1103, "*Software Handling for Capacitive Sensing*," (DS01103)

AN1104, "*Capacitive Multibutton Configurations*," (DS01104)

THE MICROCHIP WEB SITE

Microchip provides online support via our web site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

mTouch™ Sensing Solution User's Guide

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: <http://support.microchip.com>

DOCUMENT REVISION HISTORY

Revision A (September 2007)

- Initial Release of this Document.

Revision B (June 2008)

- Added more information about using custom boards

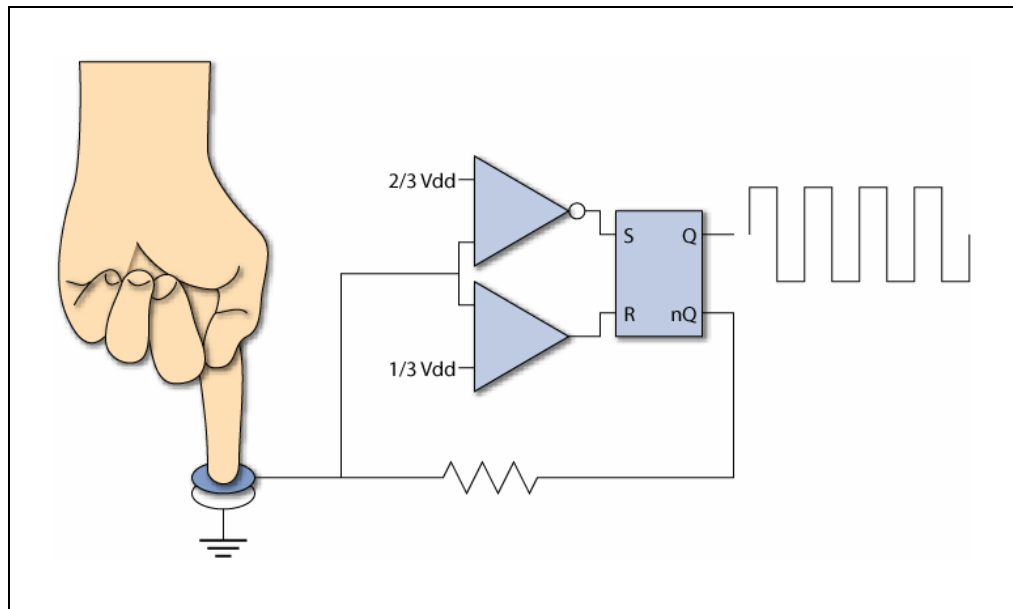
Chapter 1. Introduction

WHAT IS mTOUCH™ SENSING SOLUTION?

Microchip's mTouch™ Sensing Solution is a touch sensing solution based on changes in capacitance. Many modern applications implement capacitive sensing to provide a sleek, aesthetic, and professional look to their product.

Figure 1-1 below illustrates the basics of capacitive sensing. As a user brings their finger close to the sensing pad, additional capacitance is introduced to the system. This capacitance is detected through a PIC® microcontroller, additional circuitry and software that signals on button press.

FIGURE 1-1: BASIC ILLUSTRATION OF CAPACITIVE SENSING



The comparators and SR latch are components within the PIC® microcontroller, which allows a simple implementation of touch sense. This implementation is also open source, and there are no fees for intellectual property when using the Microchip mTouch™ Sensing Solution.

mTouch™ Sensing Solution User's Guide

INSIDE THIS USER'S GUIDE

This user's guide will aid the user in developing a design using the mTouch™ Sensing Solution. This user's guide is distributed as a part of the mTouch™ Sensing Solution Software Development Kit (SDK). It contains an explanation of what is in the SDK, which may be found at the mTouch™ Sensing Solution Design Center on www.microchip.com/mtouch.

Step-by-step guides are provided to help the user make choices particular to any design or application with regard to part selection, benefits, limitations and the necessary changes required for project specific tasks.

Development Kit Included Materials:

- mTouch™ Sensing Solution User's Guide
- Software Developer's Kit (SDK)
- Template Files
- Example Projects
- Software Applications:
 - Code Module Library v0.5 (used with SDK)
 - mTouch™ Sensing Solution Diagnostic Tool
- 4 Application Notes

The latest revisions of all software applications and application notes can be found on Microchip's web site. Current revisions are included with this release for convenience.

Lastly, while this guide and the SDK are designed to assist in the development of mTouch™ Sensing Solution applications, they are by no means the final authority on touch systems. The designer is encouraged to modify the routines in the SDK as necessary to meet the requirements of the intended application.

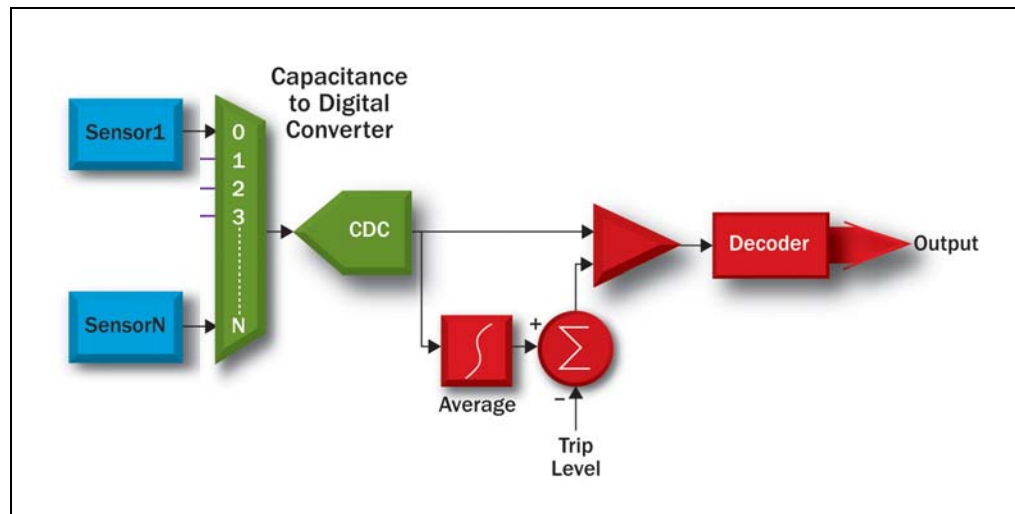
Chapter 2. How mTouch™ Sensing Solution Works

BASIC OPERATION

The mTouch™ Sensing Solution system operates by creating a parasitic capacitance between a touch sensor pad on the system PCB and ground. When the user's finger comes into close proximity to the sensor pad, a capacitance increase is generated by the iron in the user's blood. This forms an additional sensor pad and is coupled to ground through the user's extremities.

To implement a capacitive touch system, three high-level systems are required (see Figure 2-1); Sensors, Capacitance-to-Digital Converter, and Decoding. The Sensors (shown in blue) consist of the sensor pads and are multiplexed into the microcontroller. The Capacitance-to-Digital Converter (shown in green) is the hardware technique used to get a reading stored into a variable of the microcontroller. It uses internal and external hardware to create a relaxation oscillator and measure its frequency. The Decoding system (shown in red) takes the reading's value and determines what to do with it.

FIGURE 2-1: CAPACITIVE TOUCH SYSTEM



See the **Recommended Reading** section in this user's guide for more detailed information on this process.

Sensors and Capacitance-to-Digital Converter

The backbone of the physical sensing process is in creating a relaxation oscillator. The circuit in Figure 1-1 shows the hardware principle. All parts except the resistor, sensor pad and $1/3V_{DD}$ are internal to the microcontroller.

The sensor pads are conductive pads, often separated from the user's direct touch by a dielectric material. Sensor design is simple, and sensors are often simply a square or circle to provide one plate of the capacitor of about finger size to touch.

Microchip's mTouch™ Sensing Solution is based on this relaxation oscillator and uses the resistor, combined with the capacitance, to achieve an oscillation frequency in the 100's of kHz. The exact frequency is not important. The frequency is then measured through the use of Timer0 and Timer1.

Once the frequency of the oscillation has been determined, the flow of the process from Figure 2-1 has now left the sensors and Capacitance-to-Digital Converter. The sensing process then enters the Decoding system, which is software.

Decoding

The frequency reading represents the frequency of the oscillator in counts over a fixed period set by Timer0. Timer1 is used to capture the reading. Since the capacitance is unknown, a reference level for the change from nominal capacitance must be set. This is done by averaging the readings; there are two averaging methods discussed later. As a user presses the button pad, the capacitance will increase, frequency will drop and the reading will drop, too.

So, to detect a button press, we must watch for a drop in the reading by a certain amount. The amount required for the reading to drop is governed by a number of factors, but can be summarized by how sensitive the sensor is. A strong sensor will have a large drop; for example, if the user is allowed to touch the bare metal pad. A weak sensor will have a small drop in reading; for example a sensor covered by a very thick plastic. A strong sensor is better in all regards.

After the decoding of a button being scanned is complete, the reading is used in an N-point average. Typically a 16-point average is used. The average can be one of two provided methods; it may be a "gated average", or it may be a "slow average". The decrease in frequency detected is with reference to the average.

The gated average will stop averaging so that any slow environmental changes are tracked, and any rapid changes outside the "gate" are not tracked and a button press can be detected. The slow average always averages, but it does so slowly by averaging periodic values at a rate much slower than the sampling of the sensors (performed each reading).

The remaining software concerns selecting the appropriate sensor to test, configuring hardware properly and interacting with the rest of the microcontroller's duties.

Implementation Overview

A key benefit to the mTouch™ Sensing Solution is that everything may be customized! The implementation is completely open and may be rewritten at will.

The SDK provided was created using the Hi-Tech PICC™ C Compiler. Pieces of the SDK are prewritten for anticipated uses and example code for specific tasks. For example, options for the gated average and slow average exist in the SDK. Likewise, other routines are included prewritten.

Chapter 3. Building an mTouch™ Sensing Solution System

DEVELOPMENT TOOL REQUIREMENTS

Microchip's capacitive mTouch™ Sensing Solution is implemented in the C language, using Hi-Tech's PICC™ C compiler. A licensed copy or demo version may be obtained from Hi-Tech Software. The MPLAB® IDE is used with the PICC™ C compiler integrated as a language tool-suite.

To use the SDK as provided, Hi-Tech PICC™ C compiler must be used, because PICC™ C compiler specific features are used.

GETTING STARTED

This chapter describes how to use the mTouch™ Sensing Solution SDK and provided template files to implement a sensing solution. It does not provide an absolutely complete and thorough understanding of the entire process; it provides a step-by-step task-oriented approach. For a thorough understanding of the details in the sensing process, please read all the application notes found in the **Recommended Reading** section.

EXISTING SOFTWARE TEMPLATES

Starting with a template file is an easy way to begin a project. Template files are provided with the SDK that provide a template for sensing only, and a template that includes I²C™ communications for talking to the mTouch™ Sensing Solution Diagnostic Tool through a PICkit™ Serial Analyzer. The files which perform sensing only will be simpler to begin with.

These template files are filled in with an example solution. Many parts of this may be replaced with your own choices from the SDK. To use the SDK, install the Code Module Library. Within the Code Module Library program, you will need to add the snippet file "mtouch_sdk.snippet" to the list of active snippets. Within the Code Module Library, this may be done via **Settings>Preferences...** and clicking **Add File**. Browse and select the file. Its filepath should now appear in the list box. Close the Preferences dialog window, and a folder will be labeled "mTouch" in the tree to the left of the program. It contains all the pieces of the SDK. These pieces may be copied and pasted in appropriate places of the template files. A complete help file is located within the program itself, and it is also in hard copy as Appendix A in this user's guide.

SELECT A DEVICE

This SDK is designed for the three families of PIC microcontrollers in production currently capable of capacitive sensing. These families are the following:

- PIC16F610/616, PIC16HV610/616 family of microcontrollers
- PIC16F631/677/685/687/690 family of microcontrollers
- PIC16F882/883/884/886/887 family of microcontrollers

The SDK will be updated to include any new routines necessary to implement the system on new devices. Developers should routinely check Microchip's web page for updates concerning new devices and updates to the SDK.

mTouch™ Sensing Solution User's Guide

The initialization of each device family is slightly different, but accomplishes the same setup. The SDK provides an initialization routine for each device. The way the comparators, SR latch and relevant peripherals are configured is the same across parts. However, on the different parts, pinouts and registers change, which may change some of the exact values for registers.

Moving from top to bottom in the list of devices, the primary benefits are more I/O pins and more memory in the larger parts. Each part is limited to 4 capacitive buttons in a stock configuration, but in software, the smaller parts have less memory for sensing in expanded configurations as well as for your own application. The sensing channels themselves are the same on each part. So, choose a part based on what other peripherals, memory, I/O requirements are present.

EASILY CREATING SOFTWARE

The SDK is provided as template files and snippets of code; sometimes a few lines, sometimes a whole function. These snippets are contained in the file "mtouch_sdk.snippet" and may be opened with the Code Module Library application as described above in the **Existing Software Templates** section. These pieces may be spliced together in the template files provided to create software more quickly.

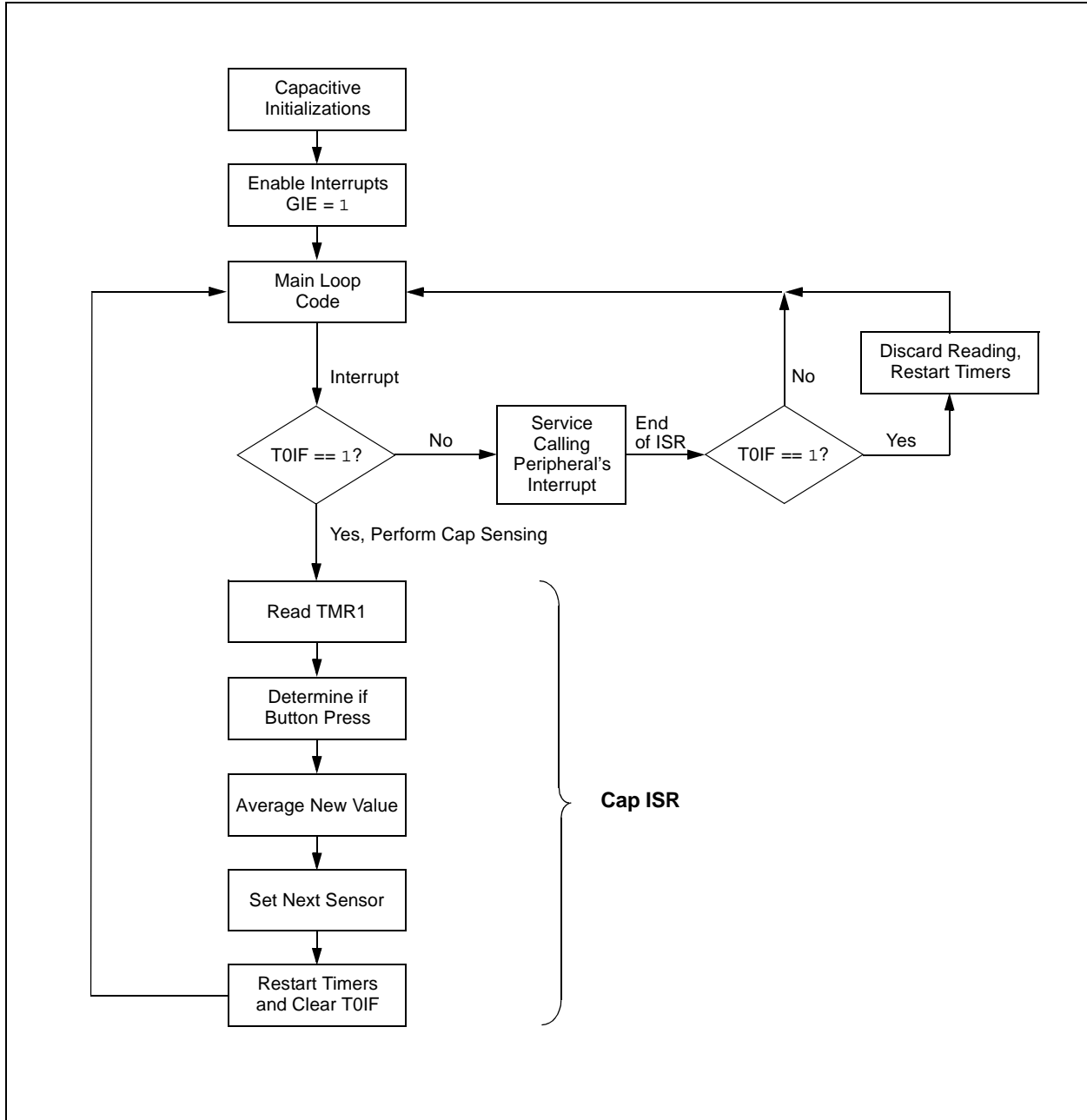
Routines have been written for a variety of different applications. Sometimes a simple negative shift of a reading is used to indicate an "On" button, sometimes that is made percentage-based, and sometimes that is further made into a press-and-release action. Proper action is application dependent, and each snippet is meant to show how to implement a certain type of action.

The Software Flowchart in Figure 3-1 shows the basic software flow of the mTouch™ Sensing Solution implementation. The flowchart illustrates the key features which the user must implement. Each block following the first "TOIF == 1?" decision block is a key block, which is provided as a snippet in the SDK. Most have additional variations on the same block. Table 3-1 shows the blocks and their associated snippets.

TABLE 3-1: RELATING FLOWCHART TO SDK SNIPPETS

Flowchart Block	Associated Snippets
Read TMR1	Reading – Get TMR1 Value
	Convert – VAL to %
	Convert – VAL to 16
Determine if Button Press	Decode – Press %
	Decode – Press 16
	Decode – Press and Release
	Decode – Percent Voting
	Decode – Paired Press
	Decode – Paired, Diff Thresh.
Average New Value	Average – Always %
	Average – Always 16
	Average – Gated %
	Average – Gated 16
Set Next Sensor	SetNextChannel 25-Button Example
	SetNextChannel 4-Button Example

FIGURE 3-1: SOFTWARE FLOWCHART



The snippets shown in the right column (Table 3-1) are not all-inclusive of what is in the SDK. The SDK provides additional software to replace nearly all blocks of the software flowchart. The user's software fits into the Main Loop Code, and may additionally fit into the Interrupt Service Routine (ISR) code, as required. A complete list of snippets, or code modules, is in **Appendix B. "Code Modules"**.

To provide a basic process to customize the template file for an application, complete the following steps.

Capacitive Initialization

1. Use `CapInit` function for the device family selected.

ISR Function

2. Select an `interrupt_isr` function to call the capacitive service routine when a capacitive interrupt occurs on Timer0 overflow. There are two ISR routines provided. The first does only capacitive sensing; the second handles an application interrupt using I²C.

Within CapISR Function

3. Use the Reading – Get TMR1 snippet to read TMR1's value.
4. Use a Convert routine. Two choices exist; using raw counts, or using percentage detection. These are labeled Convert – VAL to 16 or VAL to %. The 16 is used to indicate raw counts and its averaging scheme.
5. Place the Warm-up routine. Continue with either the 16 or % chosen before.
6. Implement Decoding for the application. This step requires a lot of customization for each application. Choose an appropriate scheme for decoding, and then use a snippet, or modify one, to suit the application if the 16 or % based detection was chosen.
7. Average in the new value. Select an averaging scheme that works with the decoding scheme.
8. Set the next sensor to scan. This sets the comparator input channels and any external I/O controlling external multiplexers if required. This section **MUST** be tailored to the application's number of buttons and hardware. A simple example and a complex example are provided in the SDK.
9. Restart Timer0 and Timer1. Call the provided function "`RestartTimers()`" at the end of the `CapISR` function. This function does not need to be modified, ever.

Variables

10. Declare and initialize all variables. This includes setting thresholds, either individually with the TRIP array, or globally with a `#define`, etc.
11. The INDEX variable must be sequenced through properly in relation to how buttons are defined. The INDEX variable relates with how the `SetNextChannel` block configures comparator inputs and any external I/O.

For a simple, stock system, it cycles 0 to 3 and then repeats. Each index value corresponds to hardware and register settings to scan a specific sensor. It is helpful to relate the index number with a word association by a `#define` in the H-file of the main file. The convention used was `iBTN0` for `i` = index, and `BTN0`, the name of the button. Customizing this with names relating to your buttons will ease development and code readability.

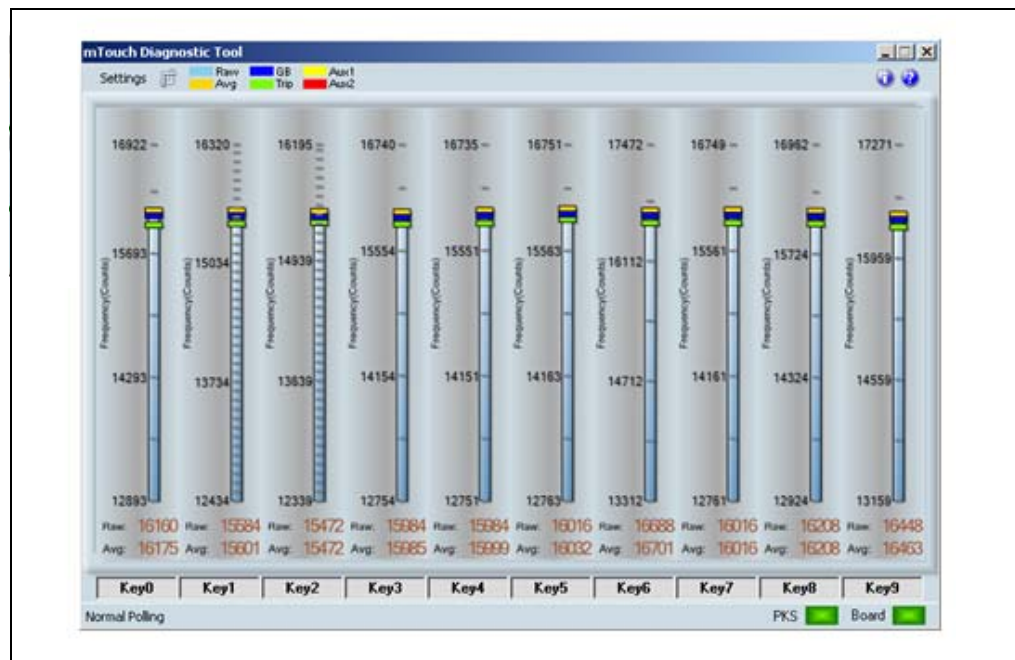
Creating the software is somewhat in-depth at first, but with a systematic approach, it is not difficult. The same key concepts must occur in each mTouch™ Sensing Solution application; so the principle may be applied repeatedly while the specific routines used within some portions may change.

SOFTWARE IN ACTION

The mTouch™ Sensing Solution Diagnostic Tool is a helpful software application provided with the developer's kit. Using a computer and a PICKit Serial Analyzer, communication between the microcontroller to the GUI exists to provide the developer an insight to the behavior of the capacitive sensors. Frequency readings are displayed in counts, and they are visually displayed to show graphically and numerically the amount of change to different factors including a user's finger press and environmental factors.

To use this tool, start with the template using I²C. The PIC microcontroller will communicate over I²C to the PICKit Serial Analyzer, and the PICKit Serial Analyzer will then communicate to the computer using USB. A screenshot of the tool in action is shown below in Figure 3-2.

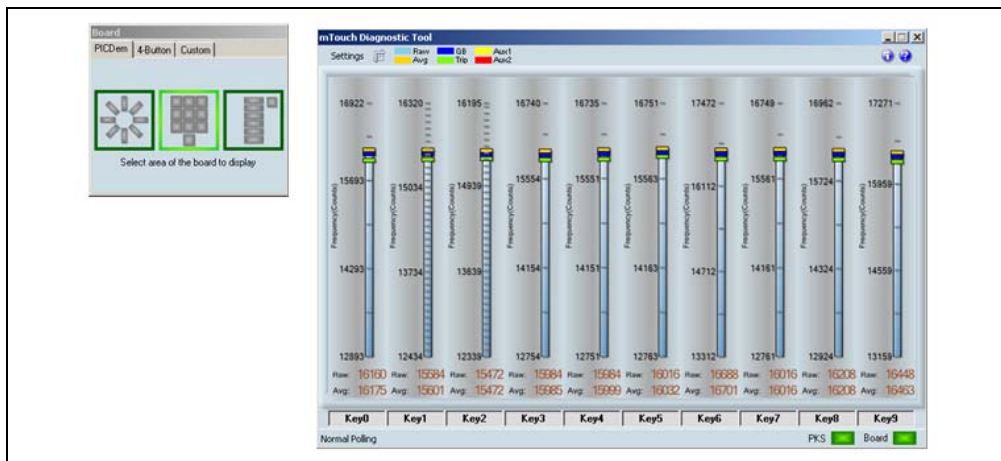
FIGURE 3-2: mTouch™ SENSING SOLUTION DIAGNOSTIC TOOL SCREENSHOT



Starting the mTouch Sensing Solution Diagnostic Tool

Start the mTouch Sensing Solution Diagnostic Tool by selecting Start>Programs>Microchip>mTouch. The mTouch Sensing Solution Diagnostic Tool main window and Board Selection window should appear as shown in Figure 3-3 when the PICKit Serial Analyzer (PKSA) is connected.

FIGURE 3-3: BOARD SELECTION AND mTouch™ SENSING SOLUTION DIAGNOSTIC TOOL WINDOWS

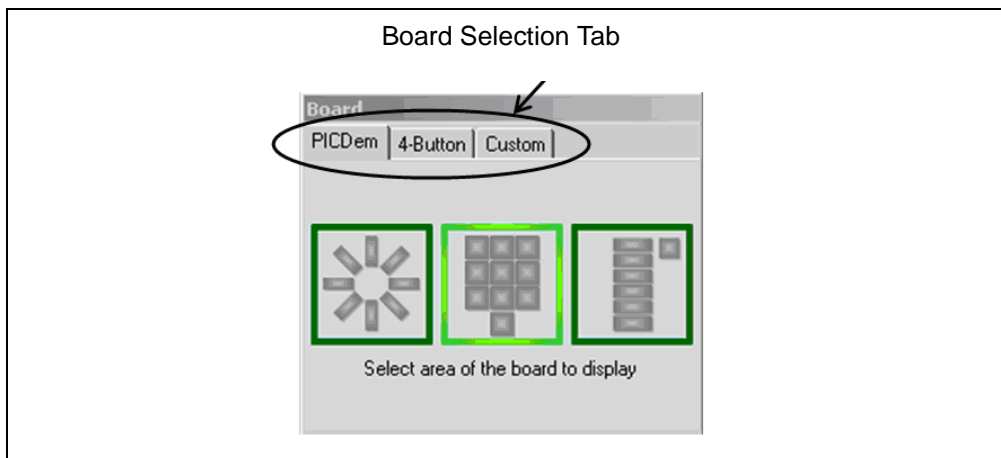


Board Selection

The Board Selection window allows user selection of three different board styles:

- PICDEM™ Touch Sense 1 Demonstration Board
- 4-Button Demonstration Board
- Custom board

FIGURE 3-4: BOARD SELECTION TAB



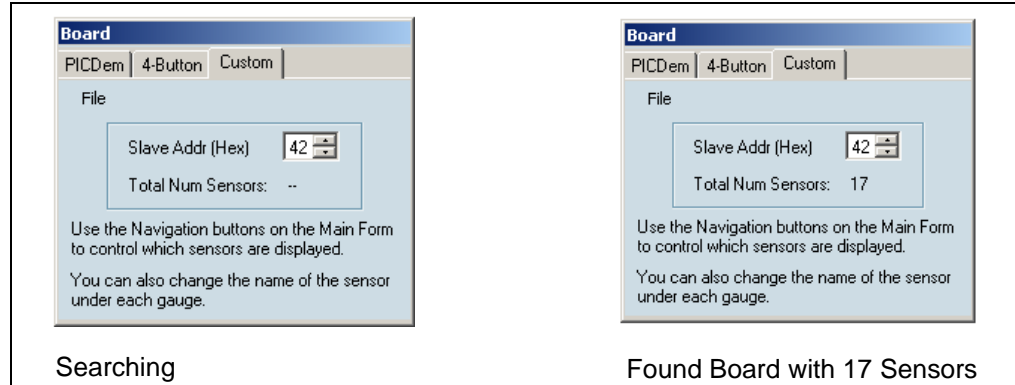
To allow communication with your custom board, the following statements found in the H-files for the project must be defined.

EXAMPLE 3-1: I²C CODE

```
#define DEVICE_ADDR    0xYY    // Device's I2C Address
#define NUM_BTNS      N        // Number of capacitive buttons
```

Figure 3-5 illustrates an example of using the mTouch Sensing Solution Diagnostic Tool Custom tab to find the board. The left dialog displays searching for the board. The right dialog displays a board and indicates how many sensors are found.

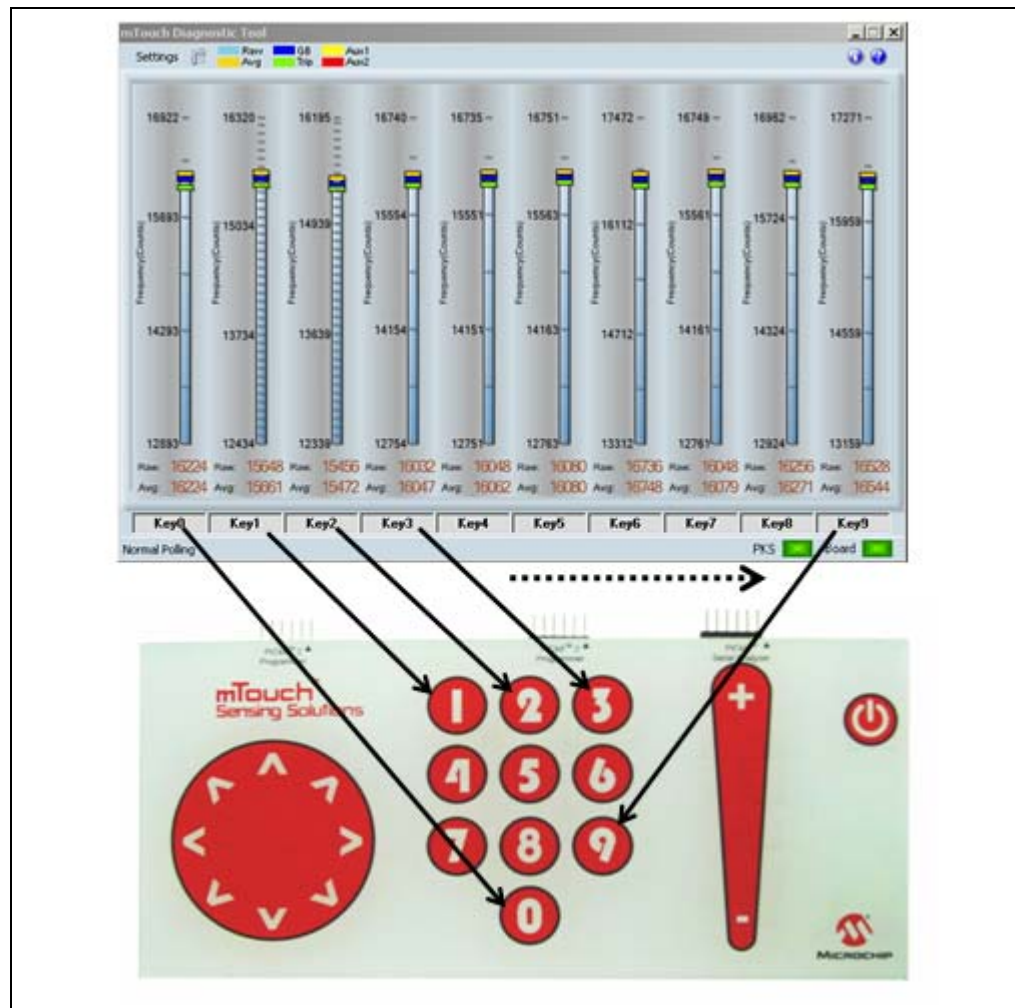
FIGURE 3-5: CUSTOM SELECTION



mTouch Sensing Solution Diagnostic Tool Main Window

Selecting a touch pad section on the Board Selection window updates the main mTouch Sensing Solution Diagnostic Tool window displaying gauges for individual touch pads in that section as shown in Figure 3-6.

**FIGURE 3-6: EXAMPLE: PICDEM™ TOUCH SENSE 1 BOARD
mTouch™ SENSING SOLUTION DIAGNOSTIC TOOL
DISPLAY AND ASSOCIATED PAD WITH KEYPAD SECTION
SELECTED IN BOARD SELECTION WINDOW**



mTouch™ Sensing Solution User's Guide

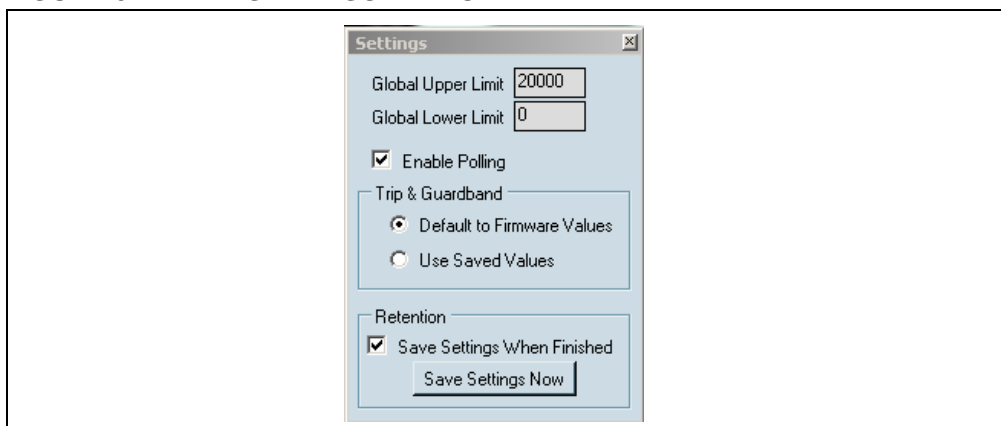
Individual gauge names can be altered from the default by clicking in the Name window directly beneath the gauge.

Select the Keypad Section of the PICDEM™ Touch Sense 1 Board in the Board Selection window. Touching the different touch pad sections, you should notice that only the Keypad in the Board Selection window shows activity. This confirms that the Demonstration Firmware Application is polling only the Keypad section and ignores the Directional Pad and Slider.

mTouch Sensing Solution Diagnostic Tool Window Configuration

In the mTouch Sensing Solution Diagnostic Tool window, select Settings. The Settings window shown in Figure 3-7 should now display.


FIGURE 3-7: SETTINGS WINDOW



In the Settings window change the value in the Global Limit window to 20000 and press the **Save Settings Now** button. The mTouch Sensing Solution Diagnostic Tool window gauges should now all be scaled to a maximum upper limit of 20000 counts.

Deselect the Enable Polling feature in the Settings window to disable software polling of the individual Keypad Keys. Touching the Keypad now has no effect on the individual gauges in the mTouch Sensing Solution Diagnostic Tool window. Select Enable Polling once again.

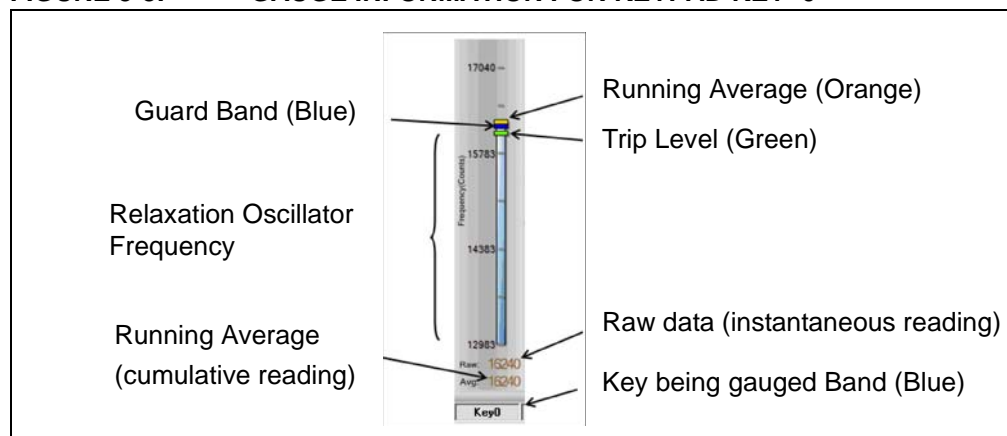
The Settings window also allows the user to save settings for the next time the tool is used.

Alternately, the **AutoScale** button  found at the top of the mTouch Sensing Solution Diagnostic Tool window can be used to scale both Upper and Lower Limits based on current average values.

mTouch Sensing Solution Diagnostic Tool Gauges

Each touch pad (Key '0' shown) on the PICDEM Touch Sense 1 Board is represented by its own individual gauge in the mTouch Sensing Solution Diagnostic Tool window. This adjustable gauge can be used to either analyze and/or change functional characteristics of the associated pad as shown in Figure 3-8.

FIGURE 3-8: GAUGE INFORMATION FOR KEYPAD KEY '0'

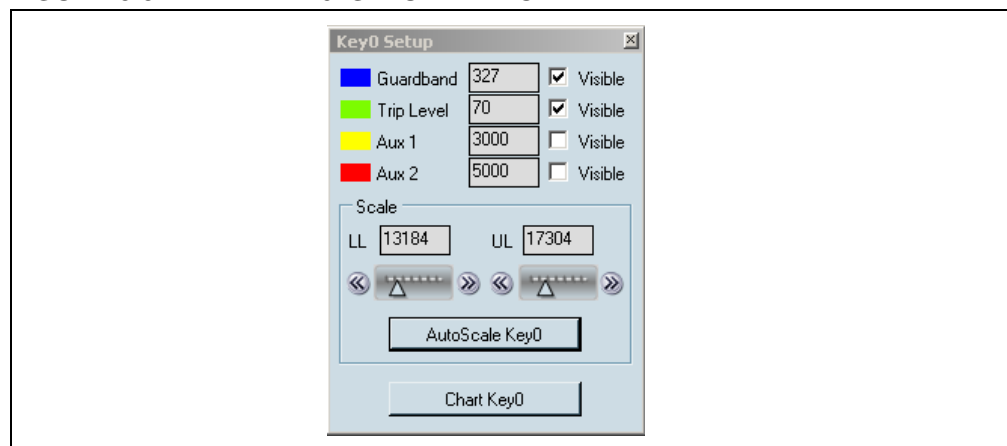


Guard Band and Trip Level parameters are easily changed by sliding their associated pointers up and down the gauge. The Guard Band provides a range of raw data values that will be included into the averaging algorithm of the demonstration software. Any raw data value below the Guard Band is not included. More information on software algorithms and other mTouch Sensing Solution Technology related topics can be found in the mTouch Sensing Solution User's Guide on the accompanying CD-ROM following installation of the mTouch Sensing Solution SDK).

Individual Gauge Configuration Using the Setup Window

Alternately, gauges can be configured using the Setup window. Click on the gauge associated with Key '0' to open the Gauge Setup window shown in Figure 3-9.

FIGURE 3-9: KEY '0' SETUP WINDOW

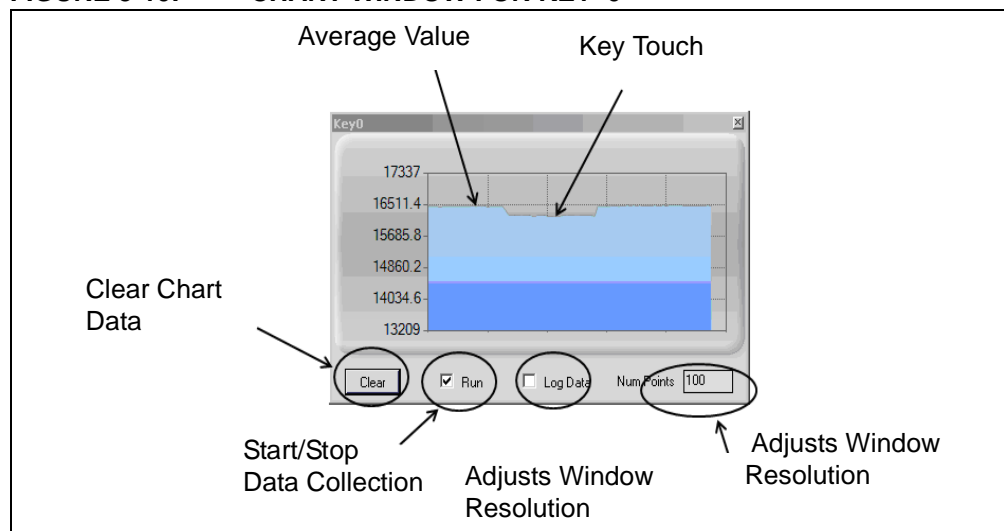


The Setup window allows customization a particular gauge by numerically adjusting parameters such as Guard Band and Trip Level. Additionally, Auxiliary pointers can be added to the gauge for other user determined references. The scale section of the window provides individual gauge scaling of both Upper (UL) and Lower Limits (LL) or by using the **AutoScale** button.

Charting Raw Data using the Chart Window

Click on the **Chart** button within the Setup window. The Chart window shown in Figure 3-10 should now be open.

FIGURE 3-10: CHART WINDOW FOR KEY '0'



The Chart window provides a visualization of the raw data for a particular touch pad as a function of time. Pressing Key '0' on the PICDEM Touch Sense 1 Board will cause the data line to dip, providing that parameters are set accordingly, as shown in Figure 3-10.

The resulting data can be stored as a .txt file within the working directory `C:\Program Files\Microchip\mTouch` and used for analytical purposes in such spreadsheet programs as Excel.

To stop recording data from the pad, deselect the check box next to Run.

The **Clear** button clears the data in the window.

Chart resolution can also be configured by changing the Num Points box value.

Directional Pad and Slider

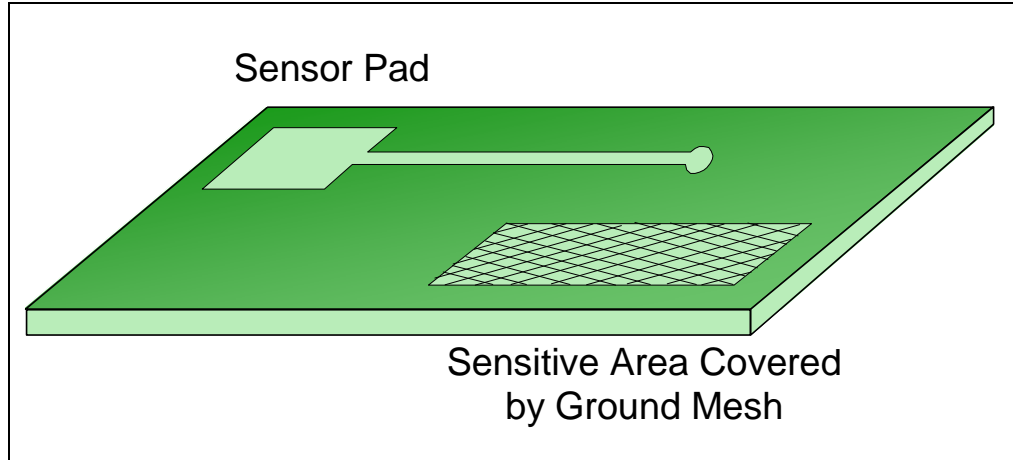
The preceding Getting Started guide is relative to both the Directional Pad and Slider section on the PICDEM Touch Sense 1 Board. The user is encouraged to complete the Getting Started guide for both these sections as well.

DEVELOPING HARDWARE

The hardware development is an important part in creating an mTouch™ sensing solution system. The design is mostly layout work, and is best described in application notes AN1102, "*Layout and Physical Design Guidelines for Capacitive Sensing*," (DS01102) and AN1104, "*Capacitive Multibutton Configurations*," (DS01104). The sensor pads are simple copper pads creating one plate of a theoretical capacitor where a finger, or conductor, creates the other plate when brought in close proximity.

The circuit on which sensor pads exist can be a PCB, flex circuit, or any conductive material. To cover any sensitive areas a lightly-meshed ground plane can be used above or around a sensitive area. Figure 3-11 shows a simple PCB drawing of this.

FIGURE 3-11: SIMPLE PCB



Attaching the PCB or sensing circuit can be done in many ways. Some ways to attach it are to use nuts and bolts, adhesives, or use the PCB itself as a touchable surface. Optically clear adhesives are good in some cases for their aesthetic appeal; simple adhesive tapes will work if the surface is hidden and the bond does not need to be aesthetic. The thinner the surface above the sensor between the user's touch, the better the sensitivity of the sensor will be.

mTouch™ Sensing Solution User's Guide

NOTES:

Chapter 4. Software Details and Choices

OVERVIEW

This chapter describes how to choose what elements of the SDK to use, since many varieties are provided. Often throughout the modules you will see some named “Snippet %” or “Snippet 16”. These names represent two different methods for detection. Those marked with a 16 use a detection based on a difference of raw counts of the frequency reading. It involves a multiplication by 16 in the process, thus the labeling 16. It is labeled to distinguish from the percentage-based routines labeled with a %. They may not be mixed and matched across the raw count to percentage-based schemes. By editing an if statement or the variables used, a snippet may be converted from one form to the other if necessary or desired.

INITIALIZATION

The initialization is an easy choice. It is dependent on which part is used, and there are three different `CapInit` functions provided for the three different PIC microcontroller families. Selection is based on the chosen microcontroller.

ISR

The `ISR` function needs little or no work to be used. Two variants are provided which provide a complete function for a sensing only, and a second variant is used to allow I²C interrupts to co-exist. This is used in the Sensing with I²C templates provided to use the mTouch™ Sensing Solution Diagnostic Tool. It is also an example to show how to implement other application interrupts without corrupting capacitive sensing readings.

TAKING A READING

The first step in taking a reading is always the same; get the TMR1 value. The value must then be converted to follow use of the either raw count detection or percentage-based detection (labeled 16 or %, respectively). Both have their merits and weaknesses.

The raw count detection is faster. It takes up less code space and it completes a scan of all buttons faster than the percentage detection. It is also very sensitive to a change of even 1 count in the frequency reading. However, it requires more careful setting and determination of trip thresholds. The mTouch™ Sensing Solution Diagnostic Tool can help with this.

The percentage-based detection offers more ease in getting a quick solution. Also, the percentage change for different buttons under similar conditions is usually very similar. So, trip thresholds are easier to define. It also allows use of the percent voting system. However, percentage-based detection requires more code in memory, because division is required and it also will delay the scan time to complete scanning all buttons.

mTouch™ Sensing Solution User's Guide

THRESHOLD DEFINITIONS

Thresholds should be defined using the raw count or percentage method, whichever is chosen. There are two threshold snippets provided:

Threshold – Absolute: Uses an array of trip values in the TRIP array.

Threshold – Percentage: Uses a single threshold for all buttons as a `#define` constant.

The percentage trip thresholds do not have to be the same across all. They may be placed in a trip array and then the same logic used with the TRIP array still applies. Whether thresholds are given the same value or individualized, place them outside of the capacitive service routine. The `CapInit` function is a good place for them.

DECODE

The decoding algorithm governs how the buttons react to a user's press. There are many varieties provided and they are by no means all that could be done. There are routines to do the following:

Press: Decodes for when a user presses down on a key

Press and Release: Decodes for a user pressing down, then releasing

Paired Press: Decodes a press of 2 unique buttons as another 3rd button

Paired Press, Diff Thresh.: Decodes a paired press using separate thresholds for the "whole" and "half" buttons

Percent Voting: Decodes using percent method, and determines the most pressed button and selects it as the pressed button

AVERAGING

Averaging should be performed after the decoding section of the `CapISR`. Two variations are provided for both the raw count and percentage-based detection. These are the gated average and the always average.

The gated average acts like a normal mechanical button would. When pressed down, the average stops tracking and the decoder always outputs a button pressed signal.

The always average is good for absolute certainty that a button will never become stuck. It slows the average down and accommodates slow environmental changes into the average, while a user's relatively fast press will trigger in the decoding section. If a finger press is held down, eventually the new reading will be averaged down to that level and the button will extinguish itself.

QUICK RELEASE

To make button presses snappy and responsive to repetitive presses, code is provided to implement a quick release of a button. It simply detects that if the reading is above the average, the button has been released. The average is then reset to the reading's value. It is provided for both the raw count and percentage-based versions.

WARM-UP

This section initializes the average values when the microcontroller is first powered up. When the device is powered up, the oscillator is not running at first and needs to establish the initial reference values.

There are two versions, one for the raw count and percentage-based detection schemes.

SETTING THE NEXT SENSOR TO TEST

The last section is a function, `SetNextChannel`, which needs to be customized for each application to match the number of buttons. It also uses the `INDEX` variable to relate the hardware configuration of the comparators and the external circuitry, if any, to be configured. Two example snippets are provided that show a simple 4-button stock system. Another shows a complex 25-button system, which uses two external multiplexers and a button going straight into the comparator.

mTouch™ Sensing Solution User's Guide

NOTES:

Chapter 5. Reference Designs

SINGLE BUTTON REFERENCE DESIGN

The Single Button Reference Design is an extremely small footprint example using the PIC16F616 QFN. It was designed to fit in a 0.8" diameter circle. On one side exists the sensor pad, and on the other all the controlling circuitry resides. This includes solder points for the programming header, power and ground, and the signal line.

The software uses a very simple decode algorithm, and it outputs a line high or low depending on the pressed or not pressed state of the sensor. An always, slowly averaging scheme is used to eliminate stuck button possibilities.

FOUR BUTTON REFERENCE DESIGN

The Four Button Reference Design is a small 1" x 3" board demonstrating the PIC16F677 in a QFN package. It is an excellent base for a simple design that also includes I²C communication to the mTouch™ Sensing Solution Diagnostic Tool. The decoding is a simple percentage-based decoding.

The size of the circuitry for capacitive sensing, excluding the pads themselves and the traces leading to the pads, is extremely small. It is approximately the same size area as on the Single Button Reference Design, except that four traces go out to sensor pads instead of a single trace and sensor pad.

MULTI-BUTTON REFERENCE DESIGN

The Multi-button Reference Design is an early demo board prototype to aid in marketing mTouch™ Sensing Solution. It includes two circuits sharing the same power rails; the two circuits feature two parts, the PIC16F616 and PIC16F887.

The PIC16F887 circuit includes a 10-button keypad, arranged in as on a cell phone, and it also includes a linear slider of 7 buttons. It uses some external multiplexers to implement all these independent sensors. In the software, the microcontroller uses an always, slow averaging scheme to eliminate the possibility of stuck buttons. It has a simple decode, but is a good circuit to base a larger scale project off of that requires external multiplexers and more complex sequencing through all of the buttons. It can also communicate to the mTouch™ Sensing Solution Diagnostic Tool (Computer GUI) to gain insight to the microcontroller's internal variables.

The PIC16F616 circuit includes 4 buttons. Three are shaped in a circular format, and one button is in the center. This circuit demonstrates a simple four button project on a lower cost microcontroller. It also uses percentage-based detection, and a voting system to ensure reliable action.

mTouch™ Sensing Solution User's Guide

NOTES:

Appendix A. Code Module Library

WHAT IS THE CODE MODULE LIBRARY?

The Code Module Library is a software productivity tool. It is intended to provide beginning and advanced software developers the ability to quickly find pieces of code which can be time consuming to recreate.

Making common building blocks easily accessible for microcontroller applications will allow developers to expedite their software development. Instead of creating a new routine to perform a software division, the user can grab an existing one in a plug-and-play sort of manner, or tweak the code to specific needs.

The Code Module Library can hold any piece of text, up to 32 kilobytes. It is intended to be used with Microchip MPLAB IDE or any other software editor.

HOW DOES THE CODE MODULE LIBRARY WORK?

Essentially, the Code Module Library is a convenient place to store your code, and then that code may be copied to the Windows® clipboard and pasted into your application where it needs to go.

Depending on what you are putting into your application, you may need to change some variables or modify the copied code. It is also very likely that you will need to declare variables used by the copied snippet in your source code in your include file or at the top of your file using "cblock" for absolute code or "res" for linked code. See the **Troubleshooting** section for more information.

Snippets are stored in snippet files with extension ".snp". The Code Module Library formats the files properly, and handles all tasks regarding creating and storing your snippets in the files. The default location for snippet files is in the Snippets folder located in the same directory where the program is installed.

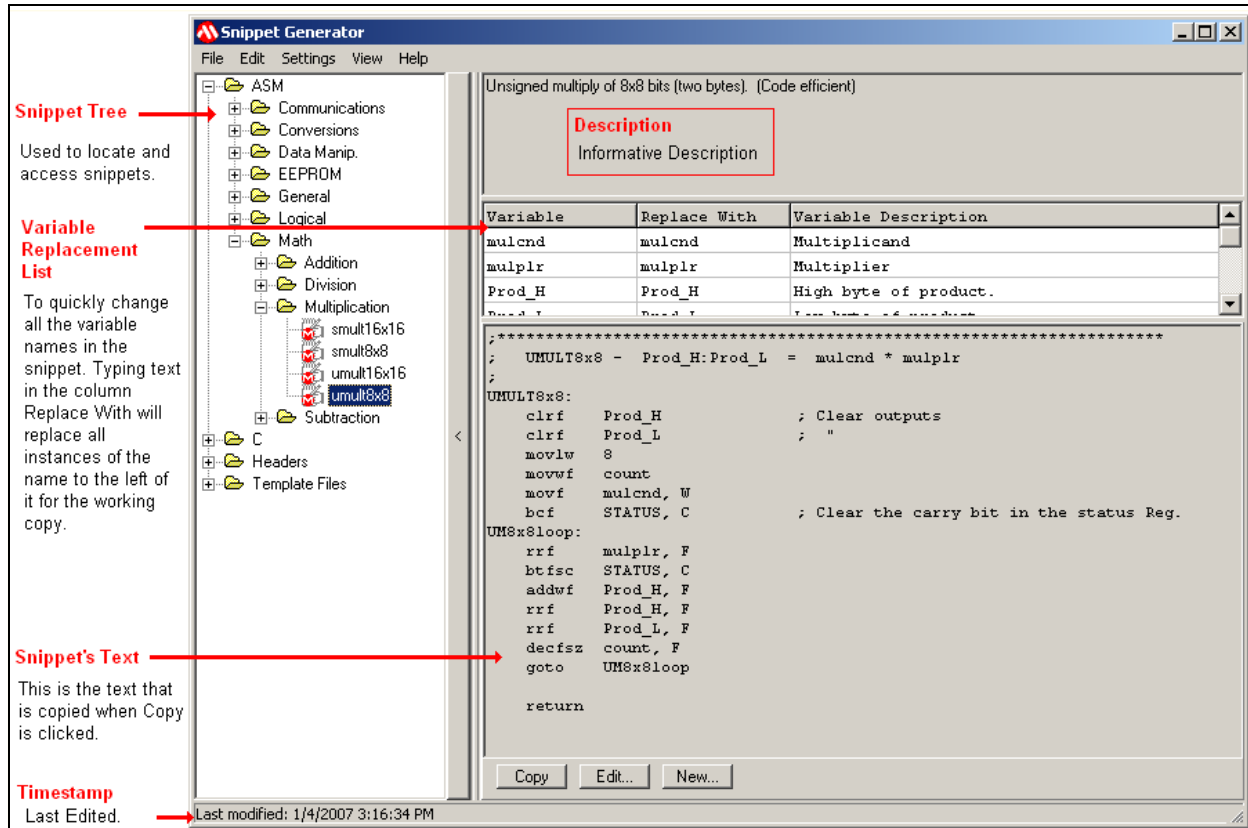
mTouch™ Sensing Solution User's Guide

INTRODUCTION TO CODE MODULE LIBRARY

This section will give you a basic explanation of what the primary components of the Code Module Library are, and how they are used.

The right pane consists of the snippet's text and some control buttons (Figure 1-1). This is the text that is copied to the clipboard when you copy a snippet. This entire pane may be hidden, leaving only the tree and the snippet's description, if so desired (snippet's text will still be copied). This way you may minimize the size of the Code Module Library on your desktop to reduce clutter. The button between the two panes will hide or show the right pane.

FIGURE 1-1: CODE MODULE LIBRARY APPLICATION WINDOW



CODE MODULE LIBRARY ITEMS

Snippet Tree

The snippet tree is your means to locate a snippet. The list of snippets is generated from a selection of files and directories under *Settings>Preferences...*. For more detailed information see “**Selecting Snippet Files Used**”. As you browse the tree, the information in the right-hand panel will refresh.

Description

This text is not copied when you copy a snippet. It is present for informative purposes and may be used to provide notes on how to use the snippet, etc. The text may be copied separately by right-clicking on the description and clicking **Copy Selected Text**.

Variable Replacement List

Some snippets require that labels or variables not be reused, particularly when using Microchip MPASM™ assembler. Labels may not be defined at two different addresses. To quickly change variable names, typing in the center column under “Replace With” will replace any instances of the text to its left with what is typed. The changes will be reflected on the screen in the working copy of the snippet, and then when copied, the snippet is copied with your variable names.

Changing the variable names under “Replace With” does not modify the snippet in the database. If you go to another snippet and return, it will refresh with the defaults. To change these variables, edit the snippet in the Edit window.

Snippet Text

This is the text that is copied when you hit ‘Ctrl+C’, click **Copy**, double-click the snippet in the Snippet Tree, or hit “Enter” twice in the Snippet Tree.

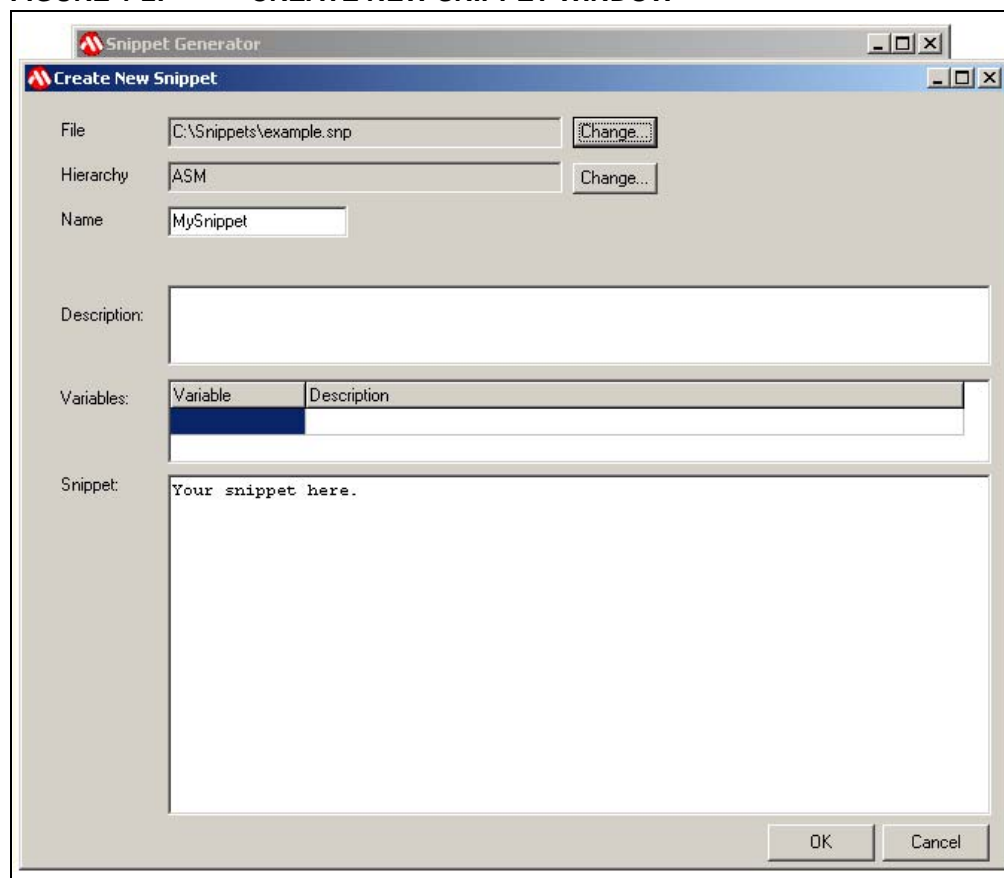
Time-stamp

The time-stamp indicates the last time that the snippet was edited.

CREATE A SNIPPET

To create a snippet, go to *File>New Snippet...* in the menu bar. Clicking **New** will bring up the window shown in Figure 1-2.

FIGURE 1-2: CREATE NEW SNIPPET WINDOW



There are three key fields to be concerned with when creating a snippet; where it will be placed on the tree is the Hierarchy field, its name, and the snippet's text at the bottom. All fields are detailed below.

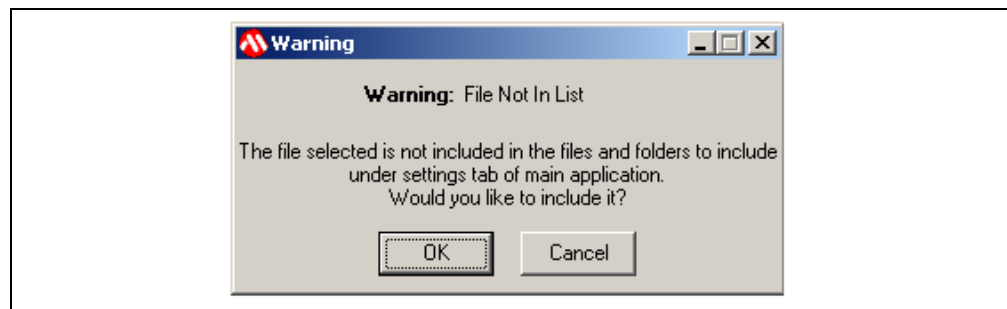
Once satisfied with what file the snippet will be located in, where in the tree it will be located, and the snippet information itself, hit **OK**, and the New window will close, create your snippet, and show it in the Main window.

Filename Field

At the top, the first grayed out box is the filename text box. This is the file where your snippet will be stored. If a snippet or directory was selected on the Snippet Tree of the Main window when you opened the New window, the New window will automatically load the file associated with the selected item. This can help keep similar snippets together in files, but to change the file, select either of the **Browse** or **New** buttons.

If you change the filename and see the window of Figure 1-2, you are trying to write the snippet in a file which is not included in the list of files and folders to display in the Snippet Tree. Everything will succeed, but if you do not click **OK**, this snippet will no longer display if you refresh your Snippet Tree. See topic “**Selecting Snippet Files Used**” for more detail.

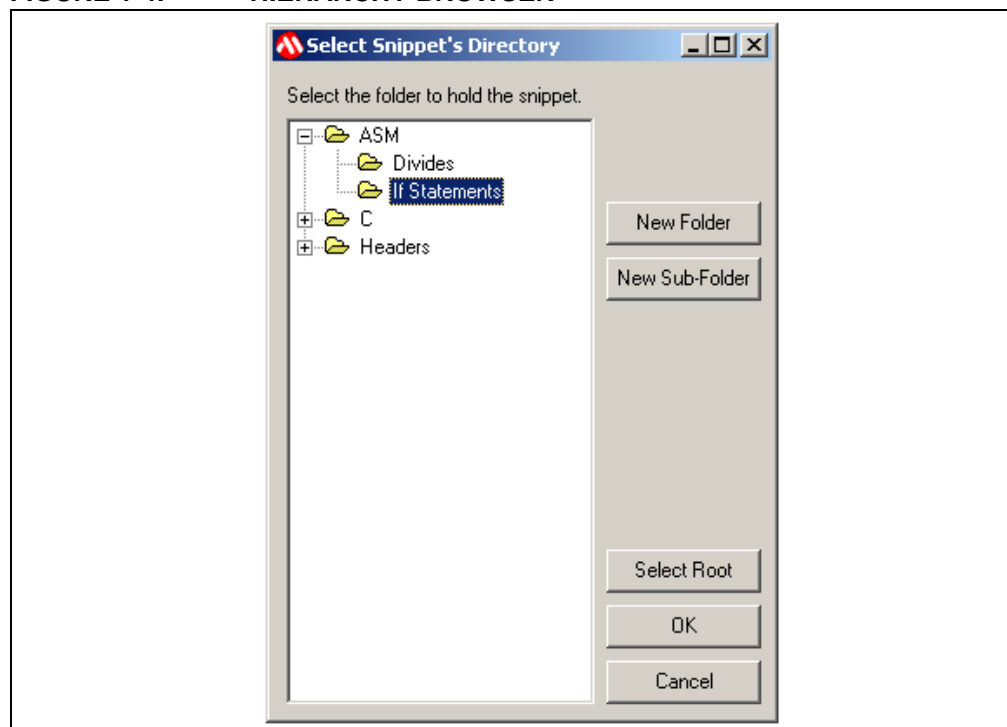
FIGURE 1-3: FILE NOT IN LIST



Hierarchy Field

This field specifies where in the Snippet Tree your snippet will be located. The Code Module Library tries to associate it where you would likely want it to go when you create a new snippet based on your selection in the Main window. To change this, and to see a more user friendly appearance, select the **Change** button. Doing so brings up the following window.

FIGURE 1-4: HIERARCHY BROWSER



mTouch™ Sensing Solution User's Guide

As seen above in Figure 1-4, if the **OK** button is clicked, the new snippet would be located under the folder If Statements, which is located under ASM, which is on the root level. The text in the hierarchy field will appear as "ASM; If Statements".

Name

Your snippet needs a name. This text entry box is where to provide the name of your snippet. Snippets in the same folder may not have the same name.

Description

The description box accompanies the snippet as seen in the Main window, and is present for a quick summary of what the snippet does or how to use it.

Variables

Variables may be declared which, when used on the Main Application window, will be replaced with text the user types under Replace With. Type in labels that will need to be changed and other variables. Even if the variable names do not need to be changed, listing them here will help identify what variables need to be declared in the user application.

Text

The text of the snippet should be typed here. This text will be your code snippet or other piece of text.

COPY A SNIPPET

Copying the snippet to the clipboard

First, a snippet item must be selected in the Snippet Tree. Snippet items display a notepad image to their left, and when selected, will cause the description and text to appear on their respective displays of the Main window.

Microchip published snippets, which have not been edited, show the Microchip logo on the lower left of the notepad icon. User snippets only have the notepad icon.

Hot Keys to Copy

Once a snippet is selected, there are several ways to copy a snippet to provide convenience for many users. The three most common means to copy a snippet are provided by accessing Edit>Copy from the menu bar, hitting Ctrl+C, or clicking the **C**opy button on the right pane.

In addition to these common ways of copying, you can also double-click a snippet to copy it, or hit 'Enter' twice while selected in the tree.

Replace Variables (Optional Step)

On the Application window, the panel shown in Figure 1-5 is located above the snippet text. A snippet may be used without replacing any words, but to quickly change instances of text under Variable, type beside it under Replace With. As shown here, `mulcnd` and `mulplr` will be replaced with user application variables `Sensor1Value` and `ScaleValue`, while `Prod_H` will be left unchanged.

FIGURE 1-5: VARIABLE REPLACEMENT LIST

Variable	Replace With	Variable Description
<code>mulcnd</code>	<code>Sensor1Value</code>	Multiplicand
<code>mulplr</code>	<code>ScaleValue</code>	Multiplier
<code>Prod_H</code>	<code>Prod_H</code>	High byte of product.

Putting the Snippet into your Application

Once the snippet is copied to the clipboard, it can be pasted into the code of your application.

Have MPLAB IDE open, or another editor you prefer, and paste the snippet where you want it to go in your editor.

Declaring Variables

After pasting a snippet into your application, if it uses variables, do not forget to declare them appropriately or you will get an error. When using MPLAB IDE and PIC microcontroller assembly, if a variable is not declared, Error 113 will occur:

“Error[113] Symbol not previously defined (VariableName)”

To fix this error, depending on if it's a variable or constant, use `cblock` or `res` for variables or use `#define` or `equ` for a constant (MPASM assembler language details can be found in MPLAB IDE Help).

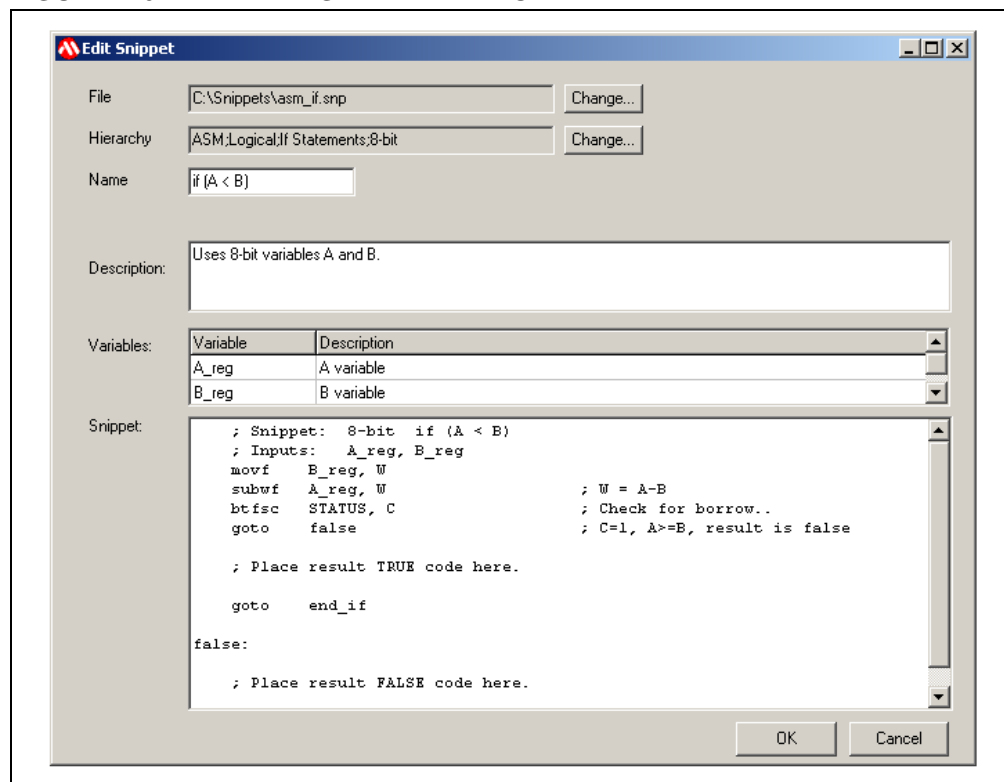
EDIT A SNIPPET

Editing a snippet is a very similar process to creating a new snippet.

First, a snippet must be selected in the tree, then click *Edit>Edit Snippet...*, or use another means to bring up the Edit window.

Below, Figure 1-6 shows the Edit window. It is styled and has the same fields as the New window, but it will replace the snippet you are editing instead of making a new one.

FIGURE 1-6: EDIT SNIPPET WINDOW



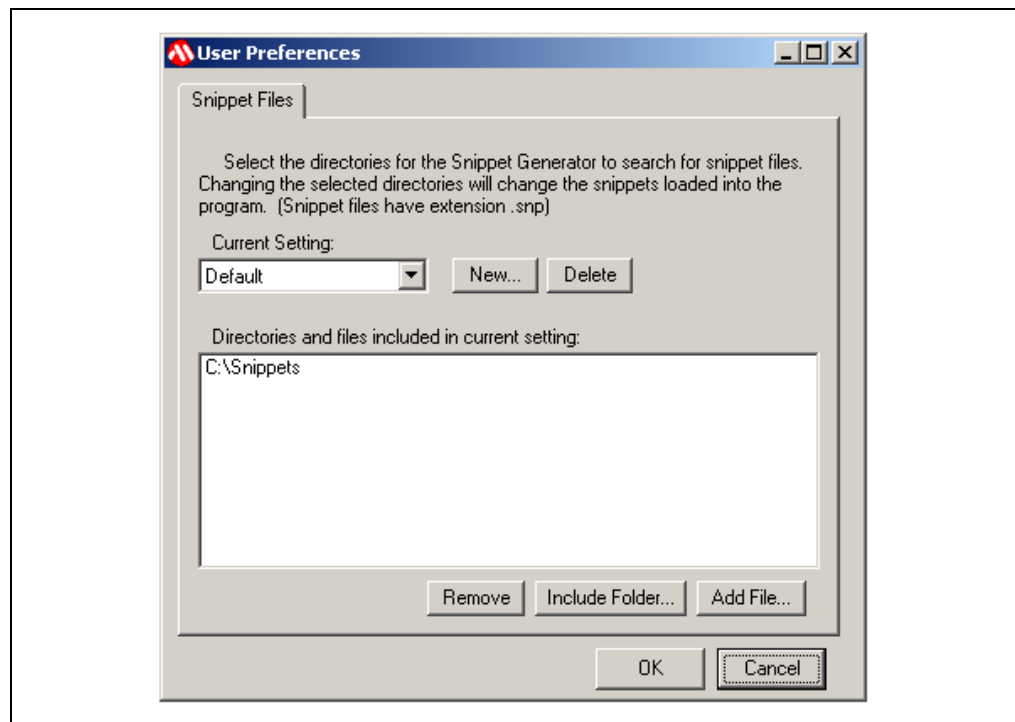
SELECTING SNIPPET FILES USED

The Code Module Library creates the Snippet Tree from a set of files. The default location and setting for these files is in the Snippets folder located where the program is installed.

It is possible to have project settings where only the snippets for a particular project may be pulled into the snippet tree. To achieve this, you can select directories which the Code Module Library should search for snippet files in, or you can specify particular snippet files to include.

To begin, go to *Settings>Preferences...*, and then a window will appear which has a tab for Snippet Files (See Figure 1-7). Under this tab you should see the Default setting in a combo box, and the files and folders included as default below that. If this has never been changed, the program will show only the Snippets folder in the directory where the program is installed.

FIGURE 1-7: USER PREFERENCES WINDOW



To create a new setting, click the **New** button to the right of the combo box. A small window will appear prompting a name to be provided (Figure 1-8). Give your new setting a name, and click **OK**. Your new setting will now be selected and a blank list will be shown below (Figure 1-9).

FIGURE 1-8: NEW SETTING NAME PROMPT

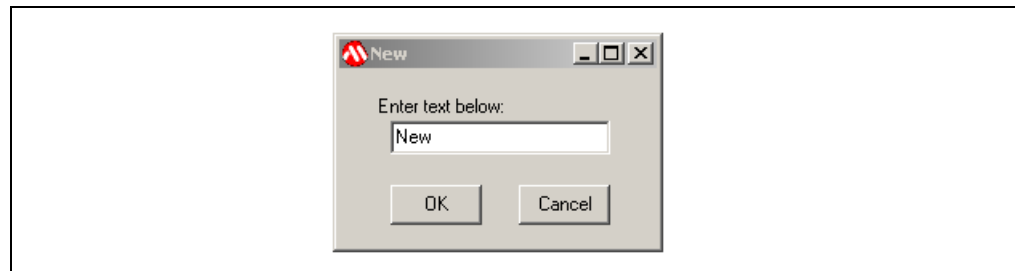
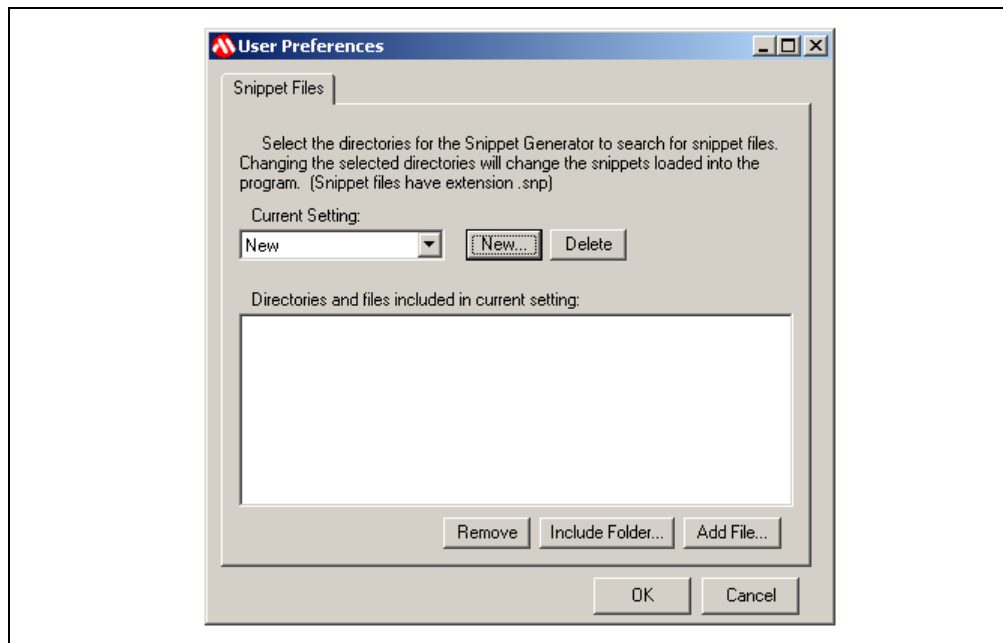


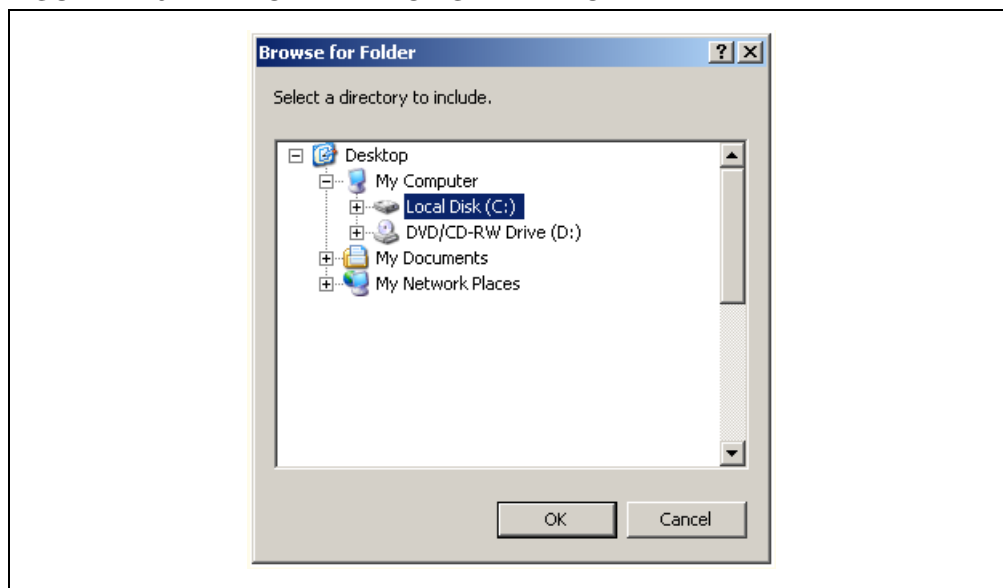
FIGURE 1-9: NEW SETTING HAS BEEN ADDED



To add a file, click **Add File** and browse for the file you wish to include. The file will appear in the list once selected.

To add a directory, click the **Include Folder** button and the window in Figure 1-10 will appear. Select the directory which contains your snippet files (.snp), and then click **OK**. This directory will be added to the list, which is non-recursive. So, adding C: (as in Figure 1-10) will only add ".snp" files directly on drive C.

FIGURE 1-10: FOLDER BROWSER WINDOW



Clicking the Preferences window's **OK** button will save your changes and load them as the current setting. The snippet tree will refresh to display all snippets found in the directories and files selected.

TIPS AND TRICKS

There are several features for convenience included in the Code Module Library.

Stay On top

The Code Module Library can Stay On Top to make it easy to go back and forth between itself and your other programs.

Hiding Snippet Text

Hiding the snippet's text will make the Code Module Library smaller and reduce clutter on your desktop. If you do not need to see the text of the snippet, this is a good option for you. Snippets can still be found through the Snippet Tree and are able to be copied without viewing the text. This also is good to use in conjunction with the Stay on Top feature to keep the Code Module Library from interfering greatly with your work environment, but quickly expanding to confirm what a snippet does or change its variables.

Directory and File Selection

Accessible through User Preferences under Settings, the directories and files used to pull your snippets from may be changed. This way, if you are working on Project A one day, and it has a lot of snippets associated with it, you can load only Project A specific snippets. Also, edited and customized snippets for a specific project can be more easily organized. Then another day, go back under Preferences and select your Project B specific snippets or use the default.

By default, the folder used will be the Snippets folder in the location where the program is installed. Any new snippets will default to the file "user.snip," but this may be changed by clicking **Change** beside the file listing in the New or Edit windows.

Quick Copying

If you browse the keyboard with your mouse, you may find it convenient to double-click a snippet name to copy it. To see all methods of copying, see section titled "**Copy A Snippet**".

TROUBLESHOOTING

Error[113] Symbol not previously defined (VariableName)

Variables and constants contained in a snippet need to be declared in your assembly code in addition to pasting the snippet. For literal constants, this does not apply (e.g., 0x42).

To fix this error, if it is a variable, declare a register for it using either “cblock” or “res” for absolute or linked code. If it is a constant (e.g., MAX_SIZE), declare it using either “#define” or “equ” followed by an appropriate number.

MPASM assembler language details and help can be found in MPLAB IDE Help.

Where can I find more snippets?

Microchip hosts a web-forum where users may share snippets and discuss issues regarding the application.

All Microchip forums may be found at: <http://forum.microchip.com/>

Bug Reporting

If you find bugs in our software, please let us know by logging onto the forum for the application labeled Bug Reports.

All Microchip forums may be found at: “<http://forum.microchip.com/>”

Appendix B. Code Modules

CODE MODULE LIST

Average – Always %
Average – Always 16
Average – Gated %
Average – Gated 16
CapInit – 61x
CapInit – 690 Family
CapInit – 88x
Convert – VAL to %
Convert – VAL to 16
Decode – Paired Press
Decode – Paired, Diff Thresh.
Decode – % Voting
Decode – Press %
Decode – Press 16
Decode – Press & Release
ISR – Basic
ISR – With I²C Calls
Quick-release %
Quick-release 16
Reading – Get TMR1 Value
RestartTimers
SetNextChannel 4-Button Example
SetNextChannel 25-Button Example
Threshold – Absolute Counts
Threshold – Percentage
Warm-up %
Warm-up 16

mTouch™ Sensing Solution User's Guide

AVERAGE – ALWAYS %

Function:	Performs averaging within a certain distance of the running average; outside of which, the averaging stops.
Hardware Dependencies:	None.
Inputs:	Variables: AVERAGE, VALUE
Outputs:	Result stored in AVERAGE.
Location:	In CapISR after decoding, before changing channels.
Options:	None.

AVERAGE – ALWAYS 16

Function:	Performs averaging of sensor data at all times, but it does so slowly. Ignoring slow changes, and counting fast changes as a press.
Hardware Dependencies:	None.
Inputs:	Variables: AVERAGE, BIGVAL, SMALLAVG
Outputs:	Result stored in AVERAGE.
Location:	After getting a raw value and converting it for appropriate scheme of detection.
Options:	None.

AVERAGE – GATED %

Function:	Performs averaging within a certain distance of the running average; outside of which, the averaging stops.
Hardware Dependencies:	None.
Inputs:	Variables: AVERAGE, VALUE
Outputs:	Configures hardware.
Location:	In CapISR after decoding, before changing channels.
Options:	None.

AVERAGE – GATED 16

Function:	Performs averaging within a certain distance of the running average; outside of which, the averaging stops.
Hardware Dependencies:	None.
Inputs:	Variables: AVERAGE, BIGVAL, SMALLAVG
Outputs:	Result stored in AVERAGE.
Location:	In CapISR after decoding, before changing channels.
Options:	None.

mTouch™ Sensing Solution User's Guide

CAPINIT 61X

Function:	Initializes PIC16F61X family cap sensing settings for the comparator registers. Uses the four C12INx- pins (x = 0-3), C2OUT, T1CKI, all else is configured digital.
Hardware Dependencies:	PIC16F610/616/PIC16FHV610/616 device.
Inputs:	Variables: RAW [], AVERAGE [], INDEX, FIRST Constants: COMP1 [], COMP2 [], NUM_BTTNS Functions: RestartTimers ()
Outputs:	Configures hardware.
Location:	CapInit () function, called from Init () .
Options:	Change analog/digital I/O per application.

CAPINIT 690 FAMILY

Function:	Initializes PIC16F690 family cap sensing settings for the comparator registers. Uses the four C12INx- pins (x = 0-3), C2OUT, T1CKI, all else is configured digital.
Hardware Dependencies:	PIC16F631/677/685/687/689/690 family device.
Inputs:	Variables: RAW [], AVERAGE [], INDEX, FIRST Constants: COMP1 [], COMP2 [], NUM_BTTNS Functions: RestartTimers ()
Outputs:	Configures hardware.
Location:	CapInit () function, called from Init () .
Options:	Change analog/digital I/O per application.

CAPINIT 88X FAMILY

Function:	Initializes PIC16F88X family cap sensing settings for the comparator registers. Uses the four C12INx- pins (X = 0-3), C2OUT, T1CKI, all else is configured digital.
Hardware Dependencies:	PIC16F882/883/884/886/887 family device.
Inputs:	Variables: RAW [], AVERAGE [], INDEX, FIRST Constants: COMP1 [], COMP2 [], NUM_BTTNS Functions: RestartTimers ()
Outputs:	Configures hardware.
Location:	CapInit () function, called from Init () .
Options:	Change analog/digital I/O per application.

mTouch™ Sensing Solution User's Guide

CONVERT VAL TO %

Function:	Converts current reading to a negative percent shift from the running average. A negative shift is made to a positive percentage, positive shifts are set to '0'.
Hardware Dependencies:	PIC16F616, PIC16F690 family, PIC16F88X family (not PIC16F610)
Inputs:	Variables: VALUE, AVERAGE []
Outputs:	Variables: percent
Location:	Place in CapISR after warm-up. It should be the 3rd item in CapISR after getting the value and the warm-up period.
Options:	In the routine multiply by 1000 to use 24.3% = 243, or multiply by 100 to use 24.3% = 24 for the percent variable computations. Multiplying by 1000 is recommended to maintain resolution without a performance loss.

CONVERT VAL TO 16

Function:	Converts VALUE to BIGVAL and SMALLAVG for detection using a drop in counts.
Hardware Dependencies:	None.
Inputs:	Variables: VALUE
Outputs:	Variables: BIGVAL, SMALLAVG
Location:	1st in CapISR routine. Afterwards, format VALUE for x16 or percentage-based detection.
Options:	None.

DECODE – PAIRED PRESS

Function:	Decodes for a paired and single presses.
Hardware Dependencies:	None.
Inputs:	Buttons structure with flags.
Outputs:	Button flags signal button detection.
Location:	After getting a raw value and converting it for appropriate scheme of detection.
Options:	None.

DECODE – PRESS, DIFF THRESH.

Function:	Uses paired presses with different thresholds for “half” and “whole” buttons.
Hardware Dependencies:	Must have paired press system.
Inputs:	Variables: TRIP [], HALFTRIP [], percent
Outputs:	Button flags signal button detection.
Location:	After getting a raw value and converting it for appropriate scheme of detection.
Options:	None.

mTouch™ Sensing Solution User's Guide

DECODE – % VOTING

Function:	Uses percentage-based detection to determine which button, if any, is the most pressed. For multiple button presses, the button with the largest change is the most pressed. Single button presses function normally.
Hardware Dependencies:	None.
Inputs:	Buttons structure with button flags, and <code>IndexArray []</code> and <code>PctArray []</code> array variables are used to sort high to low pressed values.
Outputs:	Button flags signal button detection.
Location:	After getting a raw value and converting it for appropriate scheme of detection.
Options:	None.

DECODE – PRESS %

Function:	Decodes for a press action.
Hardware Dependencies:	None.
Inputs:	Buttons structure with flags.
Outputs:	Button flags signal button detection.
Location:	After getting a raw value and converting it for appropriate scheme of detection.
Options:	None.

DECODE – PRESS 16

Function:	Decodes for a press action.
Hardware Dependencies:	None.
Inputs:	Buttons structure with flags.
Outputs:	Button flags signal button detection.
Location:	After getting a raw value and converting it for appropriate scheme of detection.
Options:	None.

DECODE – PRESS AND RELEASE

Function:	Decodes for a press and release action.
Hardware Dependencies:	None.
Inputs:	Buttons structure with flags.
Outputs:	Button flags signal button detection.
Location:	After getting a raw value and converting it for appropriate scheme of detection.
Options:	None.

mTouch™ Sensing Solution User's Guide

ISR BASIC

Function:	Responds to all interrupts and launches capacitive service routine.
Hardware Dependencies:	Possible – use caution for other interrupts as in code comments.
Inputs:	None.
Outputs:	None.
Location:	<code>interrupt isr()</code> function.
Options:	For working with other interrupts, place T0IF service first, other interrupts after it, and then at the end, check and clear T0IF if T0IF occurred during another interrupt.

ISR WITH I²C™ CALLS

Function:	Responds to all interrupts, launches capacitive service routine and services additional application interrupts.
Hardware Dependencies:	Possible – use caution for other interrupts as in code comments.
Inputs:	None.
Outputs:	None.
Location:	<code>interrupt isr()</code> function.
Options:	For working with other interrupts, place T0IF service first, other interrupts after it, and then at the end, check and clear T0IF if T0IF occurred during another interrupt.

QUICK RELEASE %

Function:	When the raw value rises above the average, a release of a press has occurred, so reset the average to the new raw value. This version for percentage-based detection code.
Hardware Dependencies:	None.
Inputs:	Variables: AVERAGE [], VALUE
Outputs:	Result stored in AVERAGE.
Location:	After averaging block in CapISR.
Options:	If doing a slowed average, could make it average each pass upward instead of a harsh jump. It makes it less prone to an error of a fluke high value.

QUICK RELEASE 16

Function:	When the raw value rises above the average, a release of a press has occurred, so reset the average to the new raw value. This version for absolute count decrease detection code.
Hardware Dependencies:	None.
Inputs:	Variables: AVERAGE [], BIGVAL
Outputs:	Result stored in AVERAGE.
Location:	After averaging block in CapISR.
Options:	If doing a slowed average, could make it average each pass upward instead of a harsh jump; makes it less prone to an error of a fluke high value.

mTouch™ Sensing Solution User's Guide

READING – GET TMR1 VALUE

Function:	Takes a reading of TMR1.
Hardware Dependencies:	None.
Inputs:	Registers: TMR1H, TMR1L
Outputs:	Variables: VALUE
Location:	1st in CapISR routine. Afterwards, format VALUE for x16 or percentage-based detection.
Options:	None.

RESTART TIMERS

Function:	Resets and Restarts TMR0 and TMR1, clears T0IF.
Hardware Dependencies:	None.
Inputs:	None.
Outputs:	None.
Location:	Stand-alone function.
Options:	None. It applies to SR latch equipped PIC microcontrollers; PIC16F88X family, PIC16F690 family and PIC16F61X family devices. Verify correctness for other microcontrollers.

SETNEXTCHANNEL 4-BUTTON EXAMPLE

Function:	Example – Switches channels between natural 4 comparator inputs.
Hardware Dependencies:	Any board using all four comparator inputs C12INx- (x = 0-3).
Inputs:	Variables: INDEX Constants: COMP1 [], COMP2 []
Outputs:	None.
Location:	As a separate function, called from CapISR before restarting timers.
Options:	Could be placed within CapISR near the end, prior to restarting the timers.

SETNEXTCHANNEL 25-BUTTON EXAMPLE

Function:	Example – Switches channels on external multiplexer and comparator inputs.
Hardware Dependencies:	Specific demo board configuration.
Inputs:	Variables: INDEX Constants: COMP1 [], COMP2 []
Outputs:	Multiplexer channel select I/O and comparator settings.
Location:	As a separate function, called from CapISR before restarting timers.
Options:	None; shows how to switch through multiplexers.

mTouch™ Sensing Solution User's Guide

THRESHOLD – ABSOLUTE COUNTS

Function:	Defining thresholds for use with detecting a drop in counts.
Hardware Dependencies:	None.
Inputs:	None.
Outputs:	None.
Location:	Place in initialization routines somewhere to set the trip values; <code>CapInit</code> would be a good location, but is not required to be placed there. The only requirement is that they must be set before the sensing routine begins.
Options:	Adjust thresholds for each project.

THRESHOLD – PERCENTAGE

Function:	Defining thresholds for use with percentage based detection.
Hardware Dependencies:	None.
Inputs:	None.
Outputs:	None.
Location:	Constants.
Options:	Change values as necessary to suit each project.

WARM-UP %

Function:	Sets up average values during short warm-up period.
Hardware Dependencies:	None.
Inputs:	Variables: FIRST, VALUE
Outputs:	Variables: None
Location:	For x16 method, place after converting to BIGVAL; for% method, place before converting VALUE to percent change.
Options:	None.

WARM-UP 16

Function:	Sets up average values during short warm-up period.
Hardware Dependencies:	None.
Inputs:	Variables: FIRST, BIGVAL
Outputs:	Variables: None
Location:	For x16 method, place after converting to BIGVAL; for% method, place before converting VALUE to percent change.
Options:	None.

mTouch™ Sensing Solution User's Guide

NOTES:



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