Capacitive Touch Algorithm Simulation

Author: Burke Davison
Microchip Technology Inc.

INTRODUCTION

This application note explains how to use the Microchip Capacitive touch Simulation Tool to customize sensing algorithms to a project's specific needs. It also helps determine the level of algorithm complexity required, based on the system's predicted environment. It assumes general knowledge of Capacitive Touch sensing and a basic understanding of the software handling process. More information on these two topics can be found in AN1101, "*Introduction to Capacitive Sensing*," and AN1103, "*Software Handling for Capacitive Sensing*".

The Microchip Capacitive Touch Simulation Tool can be downloaded from www.microchip.com.

SOFTWARE FEATURES

The provided software has been designed to accommodate several useful functions, including the generation of realistic raw sensor values and the ability to fine-tune possible sensing algorithms in order to determine the strengths and weaknesses of each. Some of these features include:

- Simulating varying speeds and directions of sensor drift
- Adding differing degrees and durations of simulated presses
- Introducing crosstalk between any two buttons so that when one is pressed another is slightly affected as well
- Adding any amount of white and conducted noise to the sensor results
- An easily exportable final result to assist in creating import files for MPLAB® Integrated Development Environment's stimulus feature or other third party analysis programs
- Five fully-implemented and customizable averaging techniques, including Fixed Threshold, Gated Average, Constant Average (Slow Average), Dual Rate Average, and Dual Average.

WINDOWS

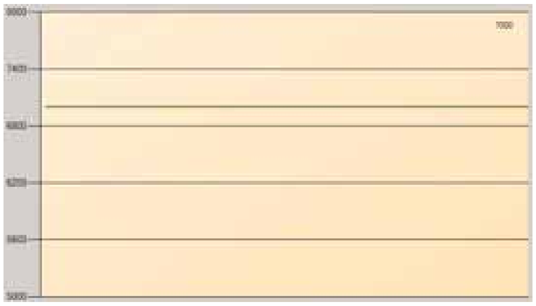
There are two windows in which user input is required to customize this application to fit a specific response. The other tabs provide information to the user on the systems performance based upon those parameters.

Setup Window

This window allows the user to specify the details behind the sensor's raw readings. The "Initial Raw Values" box at the top of the worksheet allows the user to set a starting value for each of the four simulated buttons. Realistic numbers typically range between 1,000 and 50,000.

The Microchip Capacitive Touch Simulation Tool is designed to look similar to the results that could be found by outputting the sensors' actual values during operation. Below, Figure 1 shows a blank sensor graph that is not yet using any of the provided features. Its initial starting value has been set to 7000.

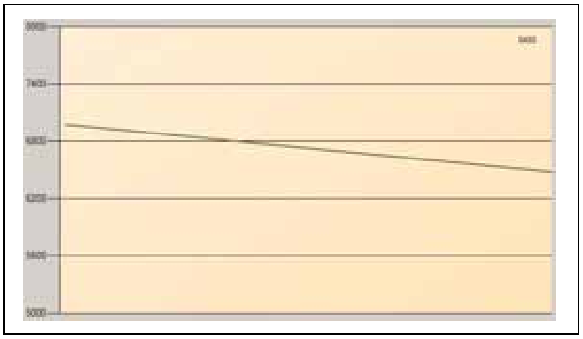
FIGURE 1: BLANK SENSOR GRAPH



DRIFT

At the bottom of the Setup window is an input field that allows the user to define a drift value for the system. This value will act as a slope for the generated sensor values. For example, a value of '-5' will cause the sensor value to decrease by 5 each time a new value is created as shown below in Figure 2.

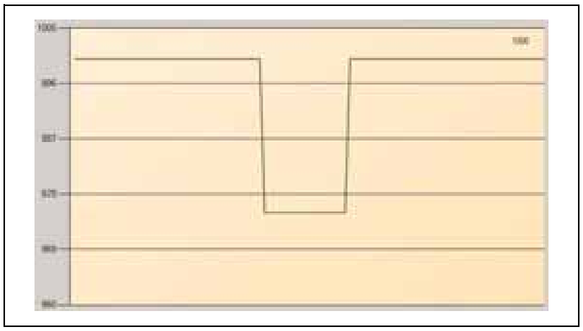
FIGURE 2: SENSOR GRAPH WITH DRIFT



PRESSES

Button press simulation is managed in two sections. The ‘Button Press %’ field determines the sensitivity, or deviation, of the press. The larger the number value, the more “firm” the press can be considered. Typical press magnitudes can range from 1% to 15%, depending on the physical design of the system. Once this value has been set, pressing the corresponding button on the main Microchip Capacitive Touch Simulation Tool window will cause the graph to simulate an actual press. This allows the user to simulate both short and long presses of varying degrees. Figure 3, below, shows a sensor with a starting value of 1000 and a 2.5% shift of 25.

FIGURE 3: SENSOR GRAPH WITH 2.5% PRESS



ADDITIONAL OPTIONS

Some additional features provide more advanced simulation choices to allow the simulations to be more realistic. These options include *Crosstalk*, *Noise*, and *Conducted Noise*.

The *Crosstalk* boxes allow the user to specify how sensitive one button is to the press-induced changes in another button. To include crosstalk in the simulation, input a percentage value in the box that corresponds to the two buttons that will be sensitive to each other. For example, if the user wanted to simulate 10% crosstalk between button 1 and button 2, a value of ‘`10`’ would be placed under the Button 1 column on the row labeled ‘Button 2 Cross Talk (%)’. This tool is not designed to simulate one-way crosstalk. Putting the value of ‘`10`’ in the box mentioned will cause changes in button 1 to effect button 2, as well as changes in button 2 to effect button 1.

The *Noise* option provides the user with a random noise source that can be added to a button’s final raw value. Adding a value to one of the boxes will cause the final raw result to deviate randomly, both above and below, by a maximum of half of that value. For example, if Button 0’s raw value had a value of `5,000` at a certain point and the noise value ‘`50`’ was inserted into the Button 0 column of this section, the final raw value would be randomly chosen between the range of 4975-5025.

The *Conducted Noise* section works exactly the same as the *Noise* section, except that it injects random noise on the final raw value only when a button is simulated as pressed. *Conducted Noise* is also the only type of noise that is transferred through cross talk.

Microchip Capacitive Touch Simulation Tool Window

This window defines the variables used to interpret the raw values to accurately determine when a button is pressed or not. In this window there are six display options, each providing the variables required for customizing the *Fixed Threshold*, *Gated Average*, *Constant Average* (or Slow Average), *Dual Rate Average*, and *Dual Average* software algorithms.

The following terminology is used in some or all of the algorithms.

THRESHOLD

The threshold is the point value at which the system decides to change state from pressed to released or from released to pressed. Valid input range is `1` to Max. Raw Value.

TRIP LIMIT

The trip limit is used to define a dynamic threshold as being a specific number of counts below an average. Valid input range is any positive integer.

SPEED OF AVERAGE

The speed of average field specifies how often the system will update the average. A value of ‘`1`’ will cause the system to update the average for every new value. A value of ‘`2`’ will update the average every other value which will result in a slower response.

AVERAGE LENGTH

The average length determines how many values are used to make the average. More accurately stated, it determines the weight each new value will have on the average with a relationship of 1/N. Its valid input range is any positive integer; however, for faster microprocessor calculation, the user should select a power of 2.

DEBOUNCE COUNTER

The debounce counter field allows the user to add debouncing to the algorithm. A value of ‘1’ will mean there is no debouncing. A value of ‘2’ will cause the system to check that two simultaneous readings agree before changing state from pressed to released or from released to pressed.

Graphs Window

When an algorithm option has been chosen in the main Microchip Capacitive Touch Simulation Tool window, the running graphs in this window will display the relevant internal variables that work to decide when a press is detected.

A binary, blue and red *detection* line has been added to the top of each graph to assist the user in determining when the system would consider a button pressed or released. On this line, high is considered pressed and low is considered released.

The y-axis displays count values, while the x-axis corresponds to time. Please note that the speed at which the graphs display new readings is much slower than the typical sensor reading speed.

Log Data

This option is provided to allow the user easy access for exporting the raw simulation values. This is available in order to help facilitate testing on the actual code through the use of the MPLAB Simulation Stimulus. To export the values, select Options – Log Data on the Graphs window, choose a name for the file, and begin the simulation. Newly generated simulation values will then be stored for later use.

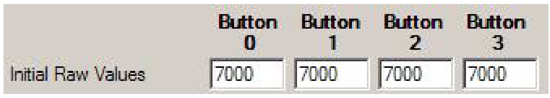
EXAMPLE SIMULATION

The following simulation shows some examples of how to use the system to fit a specific environment’s needs. This is only a theoretical example and many of the numbers have been chosen for illustration purposes alone.

Setup Window

To begin the simulation, choose a starting value for the sensor. The sensors default to 7,000, so we will use this for simplicity. In an actual system, this value is largely based on the length of capture time set up by Timer0 and the speed at which Timer1 increments. This is where a trade-off decision needs to be made between the accuracy of higher average count values and the speed at which a button press is sensed.

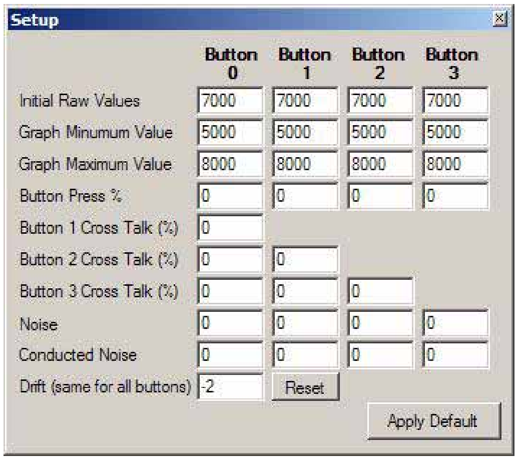
FIGURE 4: SENSOR VALUES



	Button 0	Button 1	Button 2	Button 3
Initial Raw Values	7000	7000	7000	7000

In this example, we will assume that the environment’s temperature is fluctuating slightly, which is causing a drift of -2. Figure 5 shows our starting values as we’ve currently chosen them.

FIGURE 5: DRIFT SWITCHES



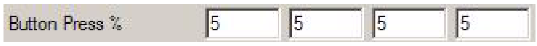
	Button 0	Button 1	Button 2	Button 3
Initial Raw Values	7000	7000	7000	7000
Graph Minumum Value	5000	5000	5000	5000
Graph Maximum Value	8000	8000	8000	8000
Button Press %	0	0	0	0
Button 1 Cross Talk (%)	0			
Button 2 Cross Talk (%)	0	0		
Button 3 Cross Talk (%)	0	0	0	
Noise	0	0	0	0
Conducted Noise	0	0	0	0
Drift (same for all buttons)	-2	Reset		

Now, using the Button Press % section of the Setup window, a shift of 5% is introduced on all of the buttons.

EQUATION 1:

$$InitialStartingValue * 0.05 = 350$$

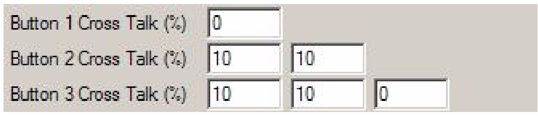
FIGURE 6: PRESS VALUES



Button Press %	5	5	5	5

Moving down to the crosstalk section, a 10% crosstalk is added between the buttons in such a way as to simulate a response similar to a 2x2 matrix. To do this, crosstalk is introduced between all buttons, except between buttons 0 and 1, and between 2 and 3.

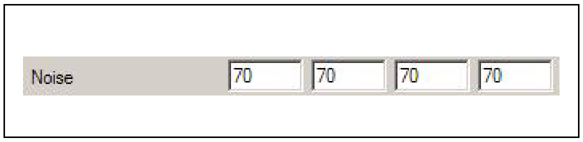
FIGURE 7: CROSSTALK VALUES



Button 1 Cross Talk (%)	0		
Button 2 Cross Talk (%)	10	10	
Button 3 Cross Talk (%)	10	10	0

In this example, we would like to create about 1% total fluctuation due to noise to simulate a decently noisy system. To achieve this, a value of ‘70’ is used in each of the *Noise* fields.

FIGURE 8: NOISE VALUES



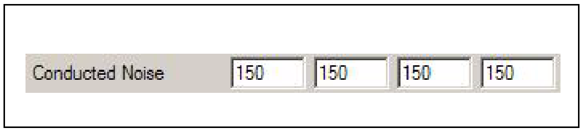
Noise 70 70 70 70

Finally, this example shows a relatively extreme case of conducted noise. A value of ‘150’ was chosen, which is equivalent to about 42.9% of the pressed amount. This value was calculated using Equation 2.

EQUATION 2:

$$\text{ConductedNoise\%} = \frac{\text{ConductedNoiseValue}}{\text{InitialStartingValue} * \text{ButtonPress\%}}$$

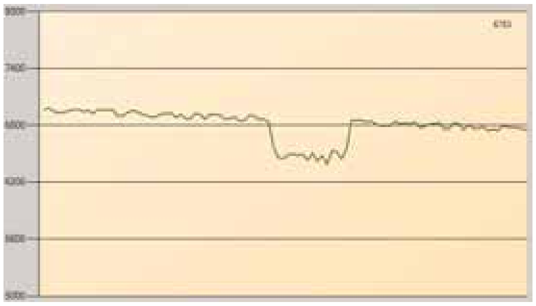
FIGURE 9: CONDUCTED NOISE VALUES



Conducted Noise 150 150 150 150

Now that all of the selections have been chosen, the Graph window will display the result. The system’s drift may cause the raw values to move beyond the graph limits. To fix this, press the **Reset** button on the Setup window or change the minimum and maximum values of the graphs. A screenshot of the generated raw graph for Button 0 is shown in Figure 10.

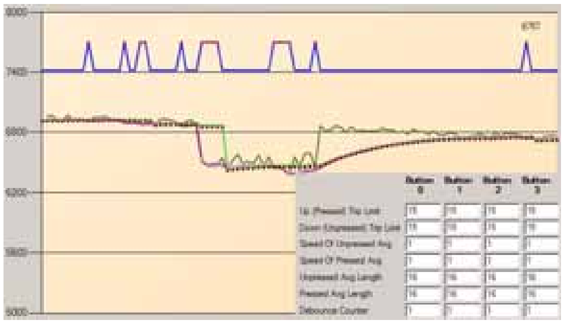
FIGURE 10: BUTTON 0 RAW VALUES



Algorithm Initialization

We will be using the *Dual Average* algorithm for this example simulation. Our initial settings for the *Dual Average* algorithm and the result can be seen in Figure 11.

FIGURE 11: ALGORITHM RESULT (INITIAL SETTINGS)

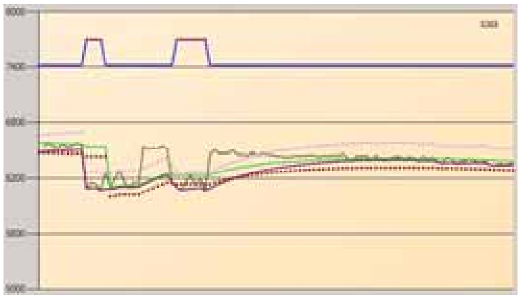


	Button 0	Button 1	Button 2	Button 3
Up (Pressed) Trip Limit	15	15	15	15
Down (Unpressed) Trip Limit	15	15	15	15
Speed Of Unpressed Avg	1	1	1	1
Speed Of Pressed Avg	1	1	1	1
Unpressed Avg Length	16	16	16	16
Pressed Avg Length	16	16	16	16
Debounce Counter	1	1	1	1

This result is clearly not desirable. The binary detection indication line at the top shows many false positives and an early release when the button is actually pressed.

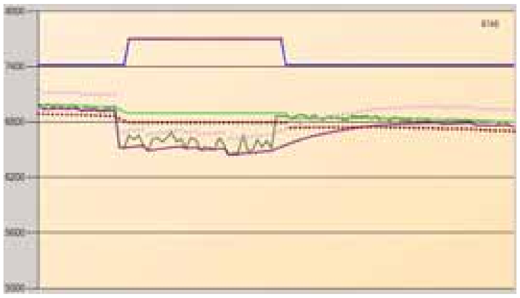
Back in the main Microchip Capacitive Touch Simulation Tool window, the trip limits are set to adjust for the noise conditions that are known to exist. In a real system, the noise levels can be determined through actual experimentation. In the example, this means the Down (Unpressed) Trip Limit is set to `100` for all buttons and the Up (Pressed) Trip Limit is set to `175` for all buttons. This seems to help the result, but does not solve all of the problems as can be seen in Figure 12.

FIGURE 12: ALGORITHM RESULT (AFTER TRIP ADJUSTMENT)



To fix these misfires and early releases, a value of `3` is placed in the Debounce counter fields for all four buttons. This will require that three captured values from the sensor agree before making a transition between the pressed and released states (Figure 13).

FIGURE 13: FINAL ALGORITHM RESULT



This final arrangement of settings seems to fit our input and environment, and may be able to act as a working solution. These simulated algorithm values can now be inserted into the actual algorithm on the product and tested.

CONCLUSION

Capacitive touch systems can be implemented using a wide range of algorithms. Choosing the correct algorithm should be based on knowledge of the operational environment and the specific requirements of the system. This simulation tool is meant to help determine a suitable algorithm by showing easily customizable simulations of possible solutions. Important application notes on designing Capacitive Touch and developing a working system are AN1103, “*Software Handling for Capacitive Sensing*”, AN1104, “*Capacitive Multibutton Configurations*”, AN1102, “*Layout and Physical Design Guidelines for Capacitive Sensing*”, and AN1101, “*Introduction to Capacitive Sensing*”.

NOTES:

APPENDIX A: ALGORITHM IMPLEMENTATIONS

Pseudo-code for the implemented algorithms are provided below for reference. Other possible implementations can be used to achieve similar or custom, design-specific results.

EXAMPLE 1: FIXED THRESHOLD

```
if ( Raw < Threshold )
    Button is pressed.
else
    Button is released.
```

EXAMPLE 2: GATED AVERAGE

```
if ( Debounce shows released )
    if ( Avg speed says to average )
        Average
    else
        Average = Prev. Average
else
    Average = Prev. Average
```

EXAMPLE 3: CONSTANT (SLOW) AVERAGE

```
if ( Avg speed says to average )
    Average
else
    Average = Prev. Average
```

EXAMPLE 4: DUAL RATE AVERAGE

```
if ( Debounce shows released )
    Average at unpressed speed
else
    Average at pressed speed
```

DUAL AVERAGE

EXAMPLE 5: UNPRESSED AVERAGE

```
if ( Debounce shows released and
   the prev. Debounce shows pressed )
    if ( Raw < Unpressed Avg )
        Unpressed Avg = Raw
    else
        Average with speed control
else
    if ( Debounce shows released )
        Average with speed control
    else
        Average = Prev. Average
```

EXAMPLE 6: PRESSED AVERAGE

```
if ( Raw < Pressed Avg OR
   (Raw < Unpressed Threshold AND
   Prev. Raw   Unpressed Threshold) )
    Pressed Avg = Raw
else
    Average with speed control
```

EXAMPLE 7: DETECTION DECISION

```
if ( Debounce shows pressed )
    if ( Raw < Unpressed Threshold )
        Button is pressed.
    else
        Button is released.
else
    if ( Raw > Pressed Threshold )
        Button is released.
    else
        Button is pressed.
```

APPENDIX B: IMPORTING TO THE MPLAB® SIMULATION STIMULUS

One of the possible uses the Microchip Capacitive Touch Simulation Tool is to create pseudo-realistic sensor values that can be imported into MPLAB and used to run simulations to discover whether or not an algorithm will be able to detect correctly. This provides both the ability to find weaknesses in algorithms and ensure the repeatability of results.

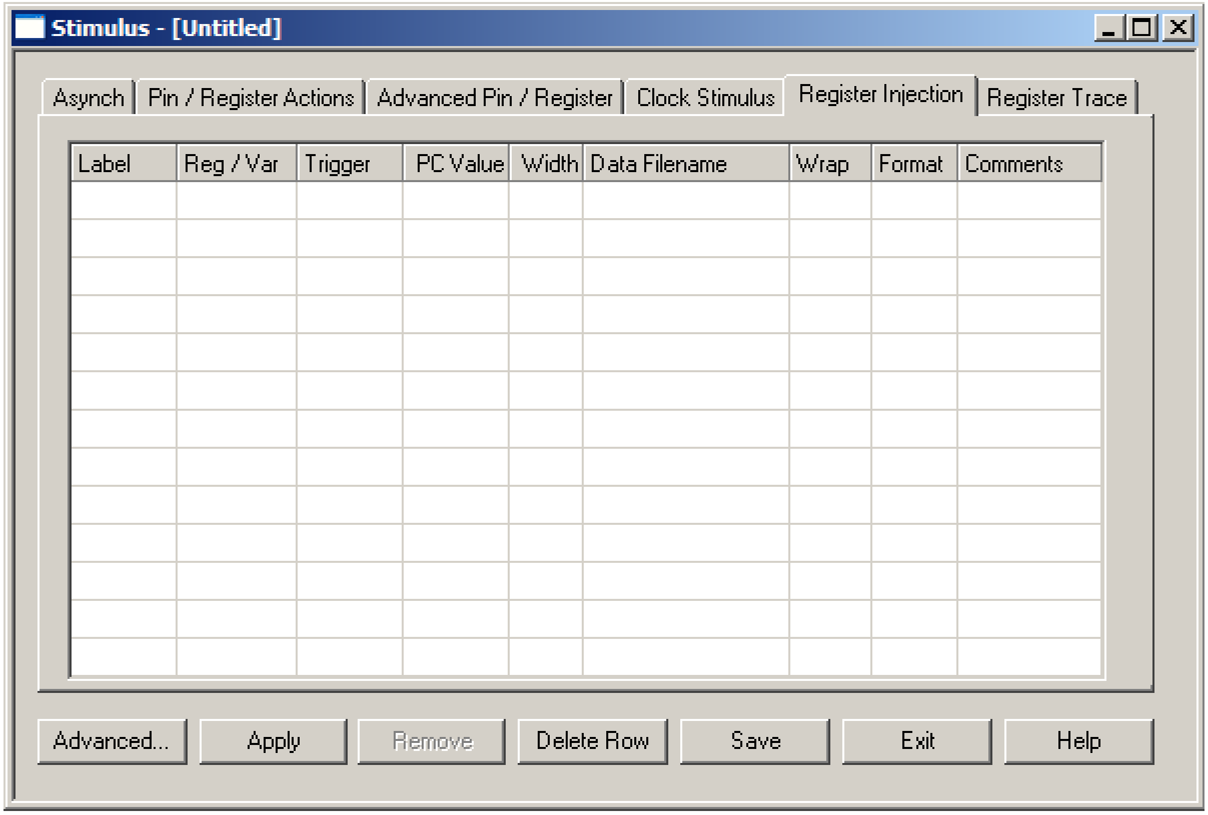
To import sensor values into a coded algorithm, first we create the values that we want to import by using the Log Data feature under the Options menu in the Graphs window. Information on how to customize the raw data values is located in the main body of this application note.

With these customized raw data parameters in place, we can save the generated results by going to Options -> Raw Data in the Graphs window. This process will save new raw values to a “`.txt`” file using a comma-delimited format. In order to prepare these values for use by the MPLAB Simulation Stimulus, the commas format for this file must be adjusted to include only the values and be separated by white space (spaces, tabs, or returns) and not commas. An easy way to do this quickly is to open the file in a simple text editor such as Microsoft® Notepad, remove the top header lines leaving only the data values, and using the Edit -> Replace feature to remove the commas.

Note: This method assumes that the PIC® MCUs algorithm is designed to work with four buttons. If the system is meant to work with fewer or more buttons than this, some additional log file editing will be required to match the system.

We now open the code in MPLAB that we want to test and go to *Debugger –> Select Tool –> MPLAB SIM* and then *Debugger –> Stimulus –> New Workbook*. Doing this opens a new Stimulus window within MPLAB that provides a wide range of functionality. For our purposes, we will only be using the **Register Injection** tab of this window which is shown in Figure B1.

FIGURE B1: STIMULUS REGISTER INJECTION WINDOW



Under the **Register Injection** tab, we click once in the white table to start a new row and then choose the fields as described below:

REG/VAR

The Reg/Var field should contain the register or variable into which the values will be injected. Please note that it is easiest to inject into a non-array variable and then have the code use that variable to inject into the array using the appropriate index value.

TRIGGER

The Trigger field defines how the stimulus decides when to perform the injection. Typically, this is set to “PC =” which defines a specific point within the code where the injection should occur.

PC VALUE

Assuming the trigger “PC =” was chosen, the PC Value field defines the PC value of the injection. Depending on the compiler that is being used, it may be possible to use the dropdown menu to define a function call as the PC value. If this is the case, it may be easier to create a dummy function `regInjection()`, which contains only a `NOP` command, and use that function as the marker for the injection. If functions are not being provided in the dropdown menu, type the absolute PC value where the injection should occur. Please note, however, that if a function is not used to define the PC Value, care must be taken after each new compile to ensure that the PC value of the desired injection location has not changed.

WIDTH

The Width field designates the number of bytes contained in each injection. In the case of this simulation, a byte width of ‘2’ should be used.

DATA FILENAME

The Data Filename field stores the location of the data file. Click once in this field and then click on the “***...***” button to browse for the `.txt` file that was previously saved.

WRAP

The Wrap field defines how the simulation will handle reaching the end of the data file. If yes, the simulation will immediately return to the beginning of the file and continue.

FORMAT

The Format field specifies in which format the numbers in the data filename are stored. In the case of this simulation, ASCII Decimal should be used.

Now that the options have been set, we click once on the **Apply** button and minimize the Stimulus window. Do not close the Stimulus window. In order for the stimulus to work, it must be open.

Note:	Remember to click the **Apply** button and do not close the stimulus. Not clicking **Apply** or closing the window will result in the stimulus not working.

Now, setting a breakpoint at the injection and running through the code using the simulation tool, we are now able to see that the variable we selected is getting injected with our generated sensor values at the PC value we specified and is able to be used by the coded algorithm as it sees fit.

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.

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

♻ 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.

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

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

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: 91-80-3090-4444
Fax: 91-80-3090-4080

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: 886-3-6578-300
Fax: 886-3-6578-370

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

03/26/09