Freescale Semiconductor
Application Note

Document Number: AN3588
Rev. 0, 01/2008

Efield Calibration Software

by: Michael Steffen
FAE North America

1 Introduction

The purpose of this application note is to explain the functionality of the software used with the efield devices MC33794, MC33940, and MC33941. The microcontroller software is responsible for calibration functionality and electrode selection in the efield devices.” The software for interfacing to the efield device is programmed into the selected microcontroller.

2 Calibration for Efield Devices

All of the calibration for these three specific efield devices is implemented in the microcontroller used to interface to the efield device. The purpose of efield device calibration is to find a baseline value to compare the value of each electrode reading. When a key is pressed on the efield device, the output voltage will be inversely proportional to the distance the finger or stimulus is near the key or electrode. Typically, the efield

Contents

device analog output “level pin #12” is connected to an analog-to-digital (A/D) input of a microcontroller. The output voltage will decrease in reference to a key press, which will decrease the A/D count inside the microcontroller.

3 Two Methods of Calibration — Static and Dynamic

Two types of calibration methods exist:

- Static calibration (or one-time calibration)
- Dynamic calibration (or adaptive calibration)

3.1 Static Calibration

Static calibration consists of taking the long-term average of all electrodes and storing the results in RAM. The averaging software can consist of taking 32 readings and computing the average.

Static calibration is typically done when the system is first powered on or during initial startup. Basically, a snapshot or capture of all the averaged A/D readings for each electrode is captured during the undisturbed (or untouched) keyboard. After the capture is complete, the averaged values are stored in memory for a comparison to a touched key or excited electrode. This is the simplest method of calibration and takes the least memory. However, the disadvantage is this algorithm is that it has to be called every time baseline values must be recomputed. This is challenging because after the system is functioning, there are no guarantees that the keyboard or system is not being disturbed or excited during recalibration. If the delta between the excited and unexcited key is small, the noise or offset drift could look like a key press or excited electrode.

3.2 Dynamic Calibration

The second and most popular method is dynamic calibration. This calibration method again consists of taking the long-term average of all electrodes and storing the results in RAM. However, the calibration is performed every X seconds, minutes, or hours depending on your preference. Use dynamic calibration to account for changing conditions that are not normal key presses or excitation. Continuously take readings at specific time intervals and use the average to use as baseline calibration values. However, be aware that this change is more of a step function or fast change.

For example, if a keyboard has been designed in applications where water or another conductive substance could be spilled on it, the system must account for this and calibrate the substance as part of the system until it can be cleaned. After the substance has been removed, the system would again recalibrate and establish another baseline. Therefore, the system would be smart enough to understand the differences between spills and key presses. With dynamic calibration, the software must look at a key press as a pulse rather than a level change. The stimulus must be recognized as a level and has a maximum time duration that is less than the calibration interval.

Dynamic calibration can be done at very fast time intervals, but you must account for this because a finger or stimulus can be calibrated into the system if it moves very slowly towards the keyboard. For example, if the system is recalibrated every 0.25 s, and then a finger moves very slowly towards the keyboard, the finger will become part of the system. This could cause a pressed key to appear as a non-pressed key because the system has calibrated the slow-moving finger as part of the keyboard.

The disadvantage in this algorithm is that the time interval must be specific to each application. You must ccount for pulse widths greater than the calibration time intervals.

3.2.1 Efield Keyboard Software Used in Demo Kits

The efield software used in the demo kits DEMO1985MC34940E and KIT33941EVM is designed to calibrate to an untouched keyboard. The VB software used in the kits allows you to select the delta that will trigger a key touch.

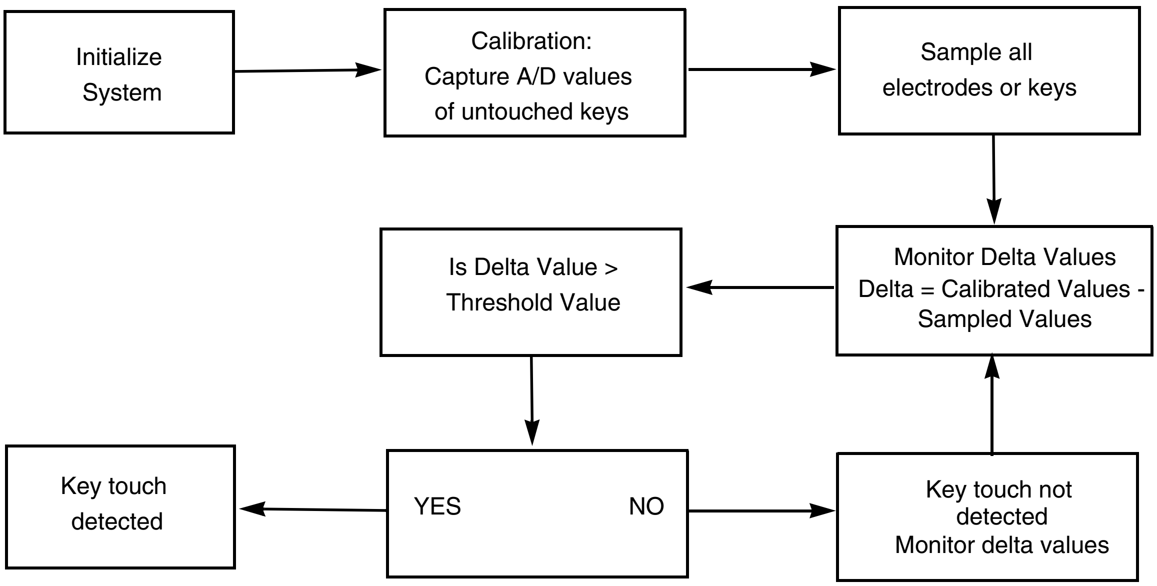

Figure 1. Efield Keyboard Software Calibration Flowchart

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3588
Rev. 0
01/2008

freescale™
semiconductor