ABSTRACT

This application report provides guidelines on how to tune capacitive touch sliders and wheels running on the MSP430™ microcontrollers. It identifies the hardware and software parameters as well as explains the steps used in tuning sliders and wheels.

The slider and wheel tuning is based on the API’s defined within the Capacitive Touch Sense Library (CAPSENSELIBRARY).

Contents

1 Overview ................................................................. 1
2 Purpose ........................................................................ 3
3 Tuning a Slider and Wheel ........................................... 3
4 Tuning Parameters .................................................... 6
5 Steps to Tuning Slider and Wheel ................................ 7
6 Conclusion ................................................................. 14
Appendix A Code for Four-Electrode Slider Using TouchPro Tool .................................................... 15

List of Figures

1 Capacitance Between Electrode and Finger ................................................................. 2
2 Linear Points Graph ......................................................................................... 3
3 Four-Electrode Slider and Three-Electrode Slider ....................................................... 4
4 Hardware Layout ............................................................................................... 5
5 Tuning Process ................................................................................................. 8
6 Three-Electrode Slider With 2-ms Gate Time and Threshold 0 ........................................ 9
7 Four-Electrode Slider ......................................................................................... 10
8 Three-Electrode Slider at 2-ms Gate Time, 16 Points, Threshold of 17 ............................ 11
9 Three-Electrode Slider at 2-ms Gate Time, 16 Points, and Threshold of 10 ....................... 12
10 Three-Electrode Slider at 2-ms Gate Time, 15 Points, and Threshold of 10 ....................... 13

1 Overview

Capacitive Touch Sensing is defined as identifying either a touch by measuring changes in capacitance. The capacitance changes due to coupling between the object doing the touching (like a human finger) and an electrically charged element or electrode.

The MSP430's capacitive touch solution is based on the Relaxation Oscillator (RO) method and fast Relaxation Oscillator (fRO) method of capacitance measurement. Electrodes are implemented using the self-capacitance technology. For more information on self-capacitance, see the Capacitive Touch Sensing, Sensor Design Guide (SLAA576).
Upon the application of a voltage to an electrode, the electrode emits an electric field in all directions, as shown in Figure 1. When a human finger comes near an electrode, the total capacitance seen by the MSP430 changes. The MSP430's algorithms monitor the changes in capacitance to determine a touch or no touch.

Figure 1. Capacitance Between Electrode and Finger

1.1 Capacitance Measurement Methods

The two capacitance measurement methods used to tune the slider and wheel were the RO and the fRO methods. Both methods were used to identify the impact of using one method over the other. In either case, the steps described in this document for tuning sliders and wheels apply to both methods.

For in-depth explanation of the two methods, see the Capacitive Touch Software Library Programmer's Guide (SLAA490) and the Capacitive Touch Sensing, Button Gate Time Optimization and Tuning Guide (SLAA574).

1.2 Types of Capacitive Touch Sensors

Capacitive touch sensors are designed as buttons, sliders, wheels and proximity sensors. The sensors are made up of one or more electrodes. The electrodes are electrically conductive materials like copper, silver, or indium tin oxide (ITO).

The difference between button, slider, wheel and proximity sensor are the size of the electrodes, the placement of the electrodes in relation to each other and the number of electrodes that make up the sensor.

This document will focus on slider and wheel types of capacitive touch sensors.

1.2.1 Sliders

A capacitive touch slider is made up of three or more electrodes placed sequentially together. When a person touches a slider, it provides back an absolute position representing the point along the slider where the touch occurred.

1.2.2 Wheels

A capacitive touch wheel is similar to a slider but differs in that the electrodes are placed in a circular layout in which the first electrode is placed adjacent to the last electrode.
2 Purpose

This user's guide is designed to help the user tune and calibrate a slider or wheel. The user will become familiar with the concepts and techniques needed to understand the different tradeoffs that exist when designing for sliders and wheels. Some examples of these concepts are response time, linearity, and resolution.

This user guide is meant to be used in conjunction with the Capacitive Touch Software Library Programmer's Guide (SLAA490) documentation and thus assumes that the user is familiar with configuring the structure.c and the terminology of the variables within this file.

The selected values for tuning variables and graphs that are listed in the following sections pertain only to the devices described in this document. The user must determine the correct variable settings for a specific device based on the PCB layout and product requirements.

3 Tuning a Slider and Wheel

Tuning is optimizing the software to provide the most linear representation of the slider or wheel's hardware response to a touch. For example, characterization of a slider's or wheel's hardware response to a touch is most likely not linear. So to compensate for this, software parameters are used to tune the hardware to provide linear sensitivity response to touch all along the entire surface of the slider or wheel.

In certain instances, even after tuning the software parameters, the hardware does not provide linearity due to its layout. In these cases, the user can decide to change the PCB layout or to write custom algorithms to compensate for the hardware design.

The software parameters used to tune the slider or wheel are the `accumulationCycles`, `maxResponse`, `threshold`, `points`, and `sensorThreshold`. The software parameters assume that the user has chosen the appropriate number of electrodes for the slider or wheel, the desired points for the slider or wheel and the inter-digitation of the electrodes making up the slider or wheel.

A slider is considered tuned when all points that are configured for the slider or wheel are detected by the software and the graph that displays the points information rises linearly from 0 to the desired number of points.

Figure 2 shows an ideal example of a points graph with a points value set to 32. In this example, the software algorithm detected every point configured for the slider and wheel as a finger or a probe slid across the slider and wheel.

![Figure 2. Linear Points Graph](image-url)
3.1 **Application Specific Tuning**

Tuning is based on the slider or wheel's intended application. Tuning a slider intended for volume control is different than tuning a slider intended for one directional swipe.

A volume control slider requires the points graph to be very linear. The sensors in the volume control slider should be designed to detect every point configured for the slider.

A slider used for directional swiping can be less linear. The tuning explained in this document attempts to achieve a linear points graph with a deviation of ±1 point.

A slider or wheel is said to provide \( n \)-bits of resolution if it can accurately distinguish between \( 2^n \) points of interest when a finger moves along its surface. For example, a 7-bit slider can accurately distinguish between 128 points of interest.

3.2 **Slider PCB Layout**

This document lists tuning results performed on a 1.25-inch slider with four electrodes, a 1.25-inch slider with three electrodes, and the 2-inch diameter wheel on the MSP430 LaunchPad BoosterPack with four electrodes. The sliders were built to be BoosterPacks for the MSP430 LaunchPad development tool per the instructions found in the LaunchPad wiki (http://processors.wiki.ti.com/index.php/BYOB).

Figure 3 shows a two-layer four-electrode slider and a two-layer three-electrode slider. The three-electrode slider has a 100% solid ground pour, while the four-electrode slider has a 25% fill hatched ground pour. The ground pours are on the top layer along with the electrodes, while the traces are on the bottom layer. The tuning described in this user guide found no differences in the tuning results when using either type of ground pour.

![Figure 3. Four-Electrode Slider and Three-Electrode Slider](image)

For a picture of the MSP430 LaunchPad BoosterPack, see the *Capacitive Touch BoosterPack (430BOOST-SENSE1)* for the LaunchPad User's Guide (SLAU337).

3.3 **Hardware Layout Used for Slider Tuning**

The sliders were built as BoosterPacks for the MSP-EXP430G2 LaunchPad using the MSP430G2553 microcontroller. A real-time data display tool, TI's TouchPro tool, was used to collect and display data sent from the LaunchPad to the PC. The Code Composer Studio™ IDE was used to download firmware to the LaunchPad and also as a debug tool.

An 8-mm diameter cylindrical copper finger probe was used to swipe across the slider and wheel (see Figure 4).
3.4 Software Application Used To Collect Data

The tuning application (see Appendix A) uses the following APIs defined within the Capacitive Touch Software Library (CAPSENSELIBRARY) to detect, set, and update the baseline value, the changes in capacitance during a touch, and to detect the position of a finger along the slider or wheel:

- TI_CAPT_Init_Baseline()
- TI_CAPT_Update_Baseline()
- TI_CAPT_Custom()
- TI_CAPT_Slider()

The application also manipulates the following values in the structure.c file (see Appendix A):

- MaxResponse
- Threshold
- Points
- sensorThreshold
- accumulationCycles
- halDefinition

3.5 Factors That Affect Tuning

There are many factors that affect tuning a slider properly to achieve the linearity specified by the application requirements. Some factors are hardware while others are software. The tuning factors are the following:

- Electrode size
- Thickness and type of overlay
- Thickness of substrate
- Slider length
- Trace length and thickness
- Interaction of electrodes within the slider
- Ground plane surrounding the slider
- Touch threshold
- Maximum Response
- Scan rate (data transmission delay, transmit buffer size, firmware execution time, total scan time for all electrodes)
- Gating Time
- Number of Position points
- Sensor threshold
- Capacitance measurement method
4 Tuning Parameters

4.1 Hardware Parameters

Part of tuning a slider or wheel also takes into consideration hardware parameters like substrate thickness, slider length, wheel diameter, type and thickness of overlay, trace lengths and thickness, electrode size, number of electrodes, inter-digitation of the electrodes within the slider or wheel, the ground plane surrounding the slider or wheel, and the placement of the slider or wheel in relation to other capacitive objects on the PCB board.

See the Capacitive Touch Sensing, Sensor Design Guide (SLAA576) for board layout designs.

4.2 Software Parameters

Software considerations for tuning a slider depends on the following parameters:

- Touch threshold
- Maximum response
- Scan rate
- Gating time
- Number of points
- Sensor Threshold
- Capacitance measurement method

For detailed explanations on the parameters, see the Capacitive Touch Software Library Programmer’s Guide (SLAA490).

4.2.1 Touch Threshold

The threshold value is the value that a change in capacitance (delta) must exceed from the baseline value before a touch is considered to be valid.

For example, in the RO measurement method, if the baseline value is 4850 counts and the threshold value is 50 counts, then a valid touch occurs when the capacitance is less than 4800 counts or a change in capacitance is greater than 50 counts.

In the fRO measurement method, if the baseline value is 4850 counts and the threshold value is 50 counts, then a valid touch occurs when the capacitance is greater than 4900 counts or a change in capacitance is greater than 50 counts.

4.2.2 Maximum Response

Maximum response parameter is used to normalize capacitive measurement in a sensor consisting of more than one electrode. This parameter helps to identify the dominant electrode within the sensor if multiple electrodes have threshold crossings during a touch. The dominant electrode is the electrode with the largest response.

4.2.3 Scan Rate

Scan Rate is the total response time of the capacitive touch system. It includes the total gate time of all capacitive touch electrodes in the system, firmware execution time, debounce and data transmission delay. Typical response times are between 7 Hz and 20 Hz; however, this range can vary depending on the system's Human Machine Interface requirements.

4.2.4 Gating Time

The gate time is the amount of time it takes to scan one electrode.
4.2.5  Position Points

The points variable defines the resolution of the slider or wheel and is used by the software algorithm to determine the exact location of a touch along the slider or wheel.

4.2.6  Sensor Threshold

Sensor Threshold variable defines the cumulative response required by the sensor to declare a valid touch and for slider or wheel, it is set to one percent.

5  Steps to Tuning Slider and Wheel

5.1  Selecting a Capacitance Measurement Method

Tuning a slider or wheel first starts with selecting a capacitance measurement method. In making the decision, the user has to understand the tradeoffs of using either method on power consumption, timer requirements and total number of electrodes to scan.

For details on the benefits and drawbacks of using either the RO or fRO method, see the Capacitive Touch Sensing, Button Gate Time Optimization and Tuning Guide (SLAA574).

A slider or wheel should be tuned every time the measurement method changes.

5.2  Selecting a Gate Time

In the RO measurement method, a fixed gate time is achieved by using the watchdog timer as the gate timer. The watchdog timer can be configured to provide gate times of 64 µs to 262ms for a 1-MHz DCO clock. The sliders and wheel described in this document used gate times of 1 ms, 2 ms, 4 ms and 8 ms. Gate times less than 1 ms resulted in delta counts of less than 50 counts, which did not allow for good distinction between a touch and no touch.

In the fRO measurement method, gate time is selected by setting the frequency oscillation cycles in the accumulationCycle variable. For a 12-MHz DCO clock, typical accumulationCycle settings for the sliders described in this document were between 300 and 2000. Settings below 200 cycles resulted in the sensor not being able to differentiate between a touch and no touch.

A slider or wheel should be tuned every time the gating time changes.

5.3  Tuning Process Flow

After the measurement method and gate times are set, the tuning process begins by first determining the values for threshold and maxResponse. Figure 5 summarizes the steps involved in tuning a slider or wheel.
Set Gate Time and HAL Definition

Set threshold and maxResponse variables to 0

Set points and sensorThreshold variables to desired values

Call TI_CAPT_Custom() function and graph result

Set threshold and maxResponse Values based on graphed result

Call TI_CAPT_Slider() function and graph result

All Points Detected?

Yes

END

No

Detection Error +/- 1 Point?

Yes

Manually verify missing points using CCS or TouchPro tool

Decrease points if possible

No

Re-Tuned?

Yes

Increase Gate Time if possible

No

Points Decreased?

Yes

Increase Gate Time?

Yes

Re-Design Slider/Wheel

No

Points

Decreased?

C

No

Yes

Re-Tuned?

D

No

Yes

Points

Decreased?

C

No

Yes

Re-Tuned?

B

No

Yes

Manually verify missing points using CCS or TouchPro tool

Decrease points if possible

Re-Tuned?

No

Set points and sensorThreshold variables to desired values

E

Figure 5. Tuning Process
The following steps describes the flowchart in Figure 5 as applied to tuning a three-electrode slider:

1. Select RO or fRO method
2. Select the clock source, clock divider and the accumulationCycle value to set the desired gate time.
3. Capture delta counts (changes in capacitance from baseline value) data for all slider electrodes.
   (a) After setting up the hardware as depicted in Figure 4 and bringing up the TouchPro real time data collection tool, set threshold value and maxResponse values, in structure.c file, to 0. This value is set for all the electrodes within the slider.
   (b) Call function TI_CAPT_Custom() in the application to get the delta counts.
   (c) Swipe across the slider at a constant pace until lobes similar to those shown in Figure 6 are seen on the display or graph.

![TouchPro Multi-Channel Graph](image)

**Figure 6. Three-Electrode Slider With 2-ms Gate Time and Threshold 0**
4. Set the **threshold** and **maxResponse** values.

   (a) **Figure 6** shows the graphed delta count values for a three electrode slider at a gating time of 2 ms per electrode and a **threshold** and **maxResponse** value set to 0.

   (b) The crossover point between electrode 0 and electrode 2 occurred at a delta count of 15, while the peak noise value occurred at a count of 9. This results in a SNR of less than 2:1. Ideally, the SNR value should be at 5:1; however, a 3:1 or 2:1 is also acceptable.

   The **maxResponse** values for each electrode occurred at 105, 103 and 118 consecutively.

   **NOTE:** If the resulting SNR is unacceptable, the user can either increase the gate time or re-layout the slider or wheel. If either is chosen, then the slider or wheel has to be re-tuned again from Step 1.

   **NOTE:** If the selected threshold value is too high, the measured value never crosses the threshold and the baseline tracking adjusts to the increase in capacitance thus, the relative capacitance measurement goes to 0 counts. One way to verify that the selected threshold is not too high is to place a finger on the slider and using a real time data display tool or CCS, verify the counts are not averaged to 0 over time.

   (c) Set the **threshold** and **maxResponse** values in structure.c file to the values determined in step 4b.

   (d) **Figure 7** shows the graphed delta counts for a four electrode slider. While setting the threshold value, the cross over point between electrode 0 and electrode 2, as well as cross over point of electrode 1 and electrode 4 should be considered.

![Figure 7. Four-Electrode Slider](image_url)
5. Tune the slider
   (a) Set the points and sensorThreshold values in structure.c file for the sensor. The points value is set based on the intended use of the slider.
   (b) Call API TI_CAPT_Slider() to determine the position of a touch along the slider.
   (c) Swipe across the slider while attempting to get a linear graph as shown in Figure 2. Figure 8 shows the actual graphed result for a 'points' value set to 16 points of interest. The resulting graph is not linear due to dead spots, scan rate and improper layout of the electrodes that resulted in discontinuities greater than ±1 point difference.

![Figure 8. Three-Electrode Slider at 2-ms Gate Time, 16 Points, Threshold of 17](image)

6. Re-capture the delta counts and decrease the threshold value and re-set maxResponse values
   (a) Repeat Step 3, but use the threshold value and maxResponse values determined in Step 4 and re-run TI_CAPT_Custom() API call.
   (b) From the resulting graph, randomly pick a threshold value that is between the original value and the new value.

   **NOTE:** A lot of variance between the original and new threshold and maxResponse values suggests that the user is swiping either more lightly or more firmly than the last time
Steps to Tuning Slider and Wheel

7. Re-tune the slider.
   (a) With the new threshold value set, re-run the `TI_CAPT_Slider()` function call to determine position points.

   Figure 9 shows a graph that lists the position points detected by the sensor. Although not linear, the graph shows that the majority of the points were detected with a detection error of ±1 point.

   **NOTE:** ‘Dead spots’ are usually found in the center of an electrode. If the points graph has a large dead spot it is because the sensitivity of the two neighboring electrodes is too low or there is not enough inter-digitation between the electrodes. The algorithm determines the dominant electrode and then uses the neighbors to position in relation to the center of the dominant electrode. If no signal is picked up from either side then it looks like a ‘dead spot’ and the value does not change until enough signal is read from either neighbor.

   Missing points in the graph occur due to a number of reasons like swiping faster than scan time, overwriting of the transmit buffer, delay in data transmission from UART channel to PC, and incorrect slider design.

![Figure 9. Three-Electrode Slider at 2-ms Gate Time, 16 Points, and Threshold of 10](image-url)
8. Decrease the points if possible
   (a) If the graph is still not linear due to large discontinuities, decrease the number of points configured for the slider or wheel if possible. If the number of points chosen is not divisible by the number of electrodes in the slider or wheel, decrease the value appropriately. Figure 10 shows the position graph for 15 points. All points configured for the slider were detected by the sensor and thus the slider can be considered tuned.

![Figure 10. Three-Electrode Slider at 2-ms Gate Time, 15 Points, and Threshold of 10](image)

9. Verify missing points.
   (a) Because the points on the graph in Figure 9 deviated ±1 point, it is a good indication that the slider is almost tuned. The final step in tuning the slider is to verify that the missing points can be detected by the sensor.
   (b) Using the TouchPro tool or Code Composer Studio IDE's watch window to monitor the position points on the slider, place a finger on or near the missing points on the slider. Without lifting the finger from the slider, move the finger around to verify that the missing point is displayed in the TouchPro tool or the watch window.
   (c) If the missing point is displayed, then the slider is tuned. If, however, the point is not displayed, decrease the number of points to be divisible by the number of electrodes in the slider.
   (d) Then re-verify the missing points using the TouchPro tool or CCS's watch window.

10. Increase the gating time
    (a) If decreasing the points and adjusting the threshold and maxResponse values do not result in a linear position graph, increase the gating time and tune the slider starting from Step 2.
6 Conclusion

A change in the gate time from 2 ms to 4 ms resulted in doubling the delta counts for capacitance measurement. At a gate time of 8 ms, the linearity of the position graph improved, as well as the SNR. For faster gate times with an improved SNR, fRO measurement method should be considered.
Appendix A Code for Four-Electrode Slider Using TouchPro Tool

A.1 Main Application

```
/* Main.c
 * Description: This program detects and displays changes in capacitance and finger positions along a 4 electrode slider designed for the MSP-EXP430G2 launchpad with the MSP430G2553 controller.
 * I/O
 *     -------------
 *   |VCC   GND|--> GND
 *   |P1.0     |
 *     Data <--|P1.1     |
 *   |P1.2     |
 *   |P1.3     |
 *   |P1.4 P1.7|
 *   |P1.5 P1.6|
 *   |P1.0 P1.7|
 *   |P2.0      P1.6|
 *    E1 <--|P2.1 P2.4|--> E4
 *    E2 <--|P2.2  P2.3|--> E3
 *     -------------
 */

#include "msp430.h"
#include "CTS_Layer.h"

void data_frame_send(void);
void UART_send_byte(unsigned char data2send);

#define Custom_Counts //If defined custom counts displayed else Slider position
#define DataHead 0x55AA //Data Header

#ifdef Custom_Counts
#define SENSOR_NUM 4 //Defines the number of data channels for display in TouchPro tool
#else
#define SENSOR_NUM 1 //Defines the 1 data channel for display for slider position
#endif

uint16_t count[SENSOR_NUM];
#define TXD BIT1 //P1.1  Bit banging this pin to send out the data to TouchPro tool
#define TXDOUT P1OUT //Output value
#define TXDDIR P1DIR //P1.1

unsigned int SliderPos = 0;

/* Function sets up the DCO and SMCLK clocks and GPIO ports */
void config(void) {
  WDTCTL = WDTPW + WDTHOLD; // Stop WDT
  BCSCTL1 = CALBC1_1MHZ; // Set range to 1MHz
  BCSCTL1 = CALDCO_1MHZ; // Set DCO to 1MHz. Declared in msp430g2553.h
  BCSCTL2 |= DIVS_2; // Set SMCLK Divider to 4 (250kHz). Sets scan time to 2.048ms. (512/250KHz)
  BCSCTL3 |= LFXT1S_2; // LFXT1 = VLO
  P1DIR  = 0xFF;
  P1OUT  = 0;
  P2DIR  = 0xFF;
  P2SEL = 0;
  P2SEL2 = 0;
  P2OUT = 0;
  TXDOUT = 0;
  TXDOUT = P1.1; //Output is low
  TXDOUT = P1.0;
  TXDOUT = P1.0;
  TXDDIR = TXD;
  TXDDIR = TXD;
  TXDDIR = TXD;
}

/* Function initializes and updates baseline for slider and monitors the changes in capacitance to determine touch or no touch. */
void main(void) {
  #ifdef Custom_Counts
  TI_CAPT_Custom(&slider, count);
  #else
  SliderPos = TI_CAPT_Slider(&slider);
  if (SliderPos == 0XFFFF) { //Set upper limit for no touch. Based on points set in structure.c
    SliderPos = 80;
  }
  count[channelIndex] = SliderPos;
  #endif
  data_frame_send(); //Send data
  while(1) {
    #ifdef Custom_Counts
    TI_CAPT_Custom(&slider, count); //Changes in capacitance
    #else
    if (SliderPos == 0XFFFF) { //Get the slider position
      SliderPos = 80;
    }
  #endif
    count[channelIndex] = SliderPos;
    data_frame_send(); //Send data
  }
```
```c
void data_frame_send(void) {
    unsigned char check_sum = 0;
    unsigned char i = 0;
    unsigned int temp = 0;
    unsigned char data_length = 0;

    data_length = SENSOR_NUM * 3 + 1; //3 channels *3 + 1 (checksum).
    temp = DataHead;
    temp >>= 8; //Send high byte
    UART_send_byte((unsigned char) temp);
    check_sum = (unsigned char) temp;
    UART_send_byte((unsigned char) DataHead);
    check_sum += (unsigned char) DataHead;
    UART_send_byte(data_length);
    check_sum += data_length;
    for (i = 0; i < SENSOR_NUM; i++) {
        UART_send_byte(i + 1);
        check_sum += i + 1;
        temp = rawCnt[i];
        temp >>= 8; //Send high byte
        UART_send_byte((unsigned char) temp);
        check_sum += (unsigned char) temp;
        UART_send_byte((unsigned char) rawCnt[i]);
        check_sum += (unsigned char) rawCnt[i];
    }
    UART_send_byte((unsigned char) check_sum);
}
```

* Function calculates data length and check sum

```c
void UART_send_byte(unsigned char data2send) {
    unsigned int temp = 0;
    unsigned int i;

    temp = data2send | (0xff00); //send low_8_bit of head
    temp <<= 1;
    for (i = BIT0; i < BITB; i <<= 1) {
        if (temp & i) {
            TXDOUT |= TXD;
        } else {
            TXDOUT &= ~TXD;
        }
        __delay_cycles(90); //Parameter for function based on clock setting
    }
    TXDOUT |= TXD; // end of one byte
}
```

* Function sends the data as low byte and high byte using bit banging method on P1.1.
* P1.1 is either cleared or set to send each value of rawCnt array
//****************************************************************************
//  RO_PINOSC_TA_WDTp example with the MSP430G2553
//  threshold and maxResponse values must be updated for electrode design,
//  system clock settings, selection of gate measurement source, and
//  accumulation cycles
//******************************************************************************

#include "structure.h"

const struct Element sliderElement1 = {
  .inputPxselRegister = (unsigned char *)&P2SEL,
  .inputPxsel2Register = (unsigned char *)&P2SEL2,
  .inputBits = BIT1,
  .maxResponse = 97,
  .threshold = 15
};

const struct Element sliderElement2 = {
  .inputPxselRegister = (unsigned char *)&P2SEL,
  .inputPxsel2Register = (unsigned char *)&P2SEL2,
  .inputBits = BIT2,
  .maxResponse = 78,
  .threshold = 15
};

const struct Element sliderElement3 = {
  .inputPxselRegister = (unsigned char *)&P2SEL,
  .inputPxsel2Register = (unsigned char *)&P2SEL2,
  .inputBits = BIT3,
  .maxResponse = 84,
  .threshold = 15
};

const struct Element sliderElement4 = {
  .inputPxselRegister = (unsigned char *)&P2SEL,
  .inputPxsel2Register = (unsigned char *)&P2SEL2,
  .inputBits = BIT4,
  .maxResponse = 89,
  .threshold = 15
};

const struct Sensor slider = {
  .halDefinition = RO_PINOSC_TA0_WDTp,
  .numElements = 4, //total number of elements that make up the slider
  .baseOffset = 0,  //is a cumulative count of the number of elements defined in this application.
  .measGateSource = GATE_WDT_SMCLK, // 0->SMCLK, 1-> ACLK
  .accumulationCycles = WDTp_GATE_512, //512
  .points = 32,
  .sensorThreshold = 1
};
# IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have not been so designated is solely at the Buyer’s risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

## Products

<table>
<thead>
<tr>
<th>Audio</th>
<th><a href="http://www.ti.com/audio">www.ti.com/audio</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Amplifiers</td>
<td>amplifier.ti.com</td>
</tr>
<tr>
<td>Data Converters</td>
<td>dataconverter.ti.com</td>
</tr>
<tr>
<td>DLP® Products</td>
<td><a href="http://www.dlp.com">www.dlp.com</a></td>
</tr>
<tr>
<td>DSP</td>
<td>dsp.ti.com</td>
</tr>
<tr>
<td>Clocks and Timers</td>
<td><a href="http://www.ti.com/clocks">www.ti.com/clocks</a></td>
</tr>
<tr>
<td>Interface</td>
<td>interface.ti.com</td>
</tr>
<tr>
<td>Logic</td>
<td>logic.ti.com</td>
</tr>
<tr>
<td>Power Mgmt</td>
<td>power.ti.com</td>
</tr>
<tr>
<td>Microcontrollers</td>
<td>microcontroller.ti.com</td>
</tr>
<tr>
<td>RFID</td>
<td><a href="http://www.ti-rfid.com">www.ti-rfid.com</a></td>
</tr>
<tr>
<td>OMAP Applications Processors</td>
<td><a href="http://www.ti.com/omap">www.ti.com/omap</a></td>
</tr>
<tr>
<td>Wireless Connectivity</td>
<td><a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a></td>
</tr>
<tr>
<td>Applications</td>
<td>Automotive and Transportation <a href="http://www.ti.com/automotive">www.ti.com/automotive</a></td>
</tr>
<tr>
<td></td>
<td>Communications and Telecom <a href="http://www.ti.com/communications">www.ti.com/communications</a></td>
</tr>
<tr>
<td></td>
<td>Computers and Peripherals <a href="http://www.ti.com/computers">www.ti.com/computers</a></td>
</tr>
<tr>
<td></td>
<td>Consumer Electronics <a href="http://www.ti.com/consumer-apps">www.ti.com/consumer-apps</a></td>
</tr>
<tr>
<td></td>
<td>Energy and Lighting <a href="http://www.ti.com/energy">www.ti.com/energy</a></td>
</tr>
<tr>
<td></td>
<td>Industrial <a href="http://www.ti.com/industrial">www.ti.com/industrial</a></td>
</tr>
<tr>
<td></td>
<td>Medical <a href="http://www.ti.com/medical">www.ti.com/medical</a></td>
</tr>
<tr>
<td></td>
<td>Security <a href="http://www.ti.com/security">www.ti.com/security</a></td>
</tr>
<tr>
<td></td>
<td>Space, Avionics and Defense <a href="http://www.ti.com/space-avionics-defense">www.ti.com/space-avionics-defense</a></td>
</tr>
<tr>
<td></td>
<td>Video and Imaging <a href="http://www.ti.com/video">www.ti.com/video</a></td>
</tr>
</tbody>
</table>

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2013, Texas Instruments Incorporated