

## **Getting Started With the Capacitive Touch Software Library (CAPSENSELIBRARY)**

---

---

---

This document explains the process of getting started with the [Capacitive Touch Software Library](#). There are multiple ways to perform capacitive touch sensing with the MSP430™ microcontrollers, and this document gives an overview of the methods available, the applicable target platforms, and example projects to start development. The example projects are accompanied by a step-by-step walkthrough of library configuration with the Capacitive Touch BoosterPack™ (430BOOST-SENSE1) based on the MSP430 LaunchPad™ Value Line Development Kit (MSP-EXP430G2) using the MSP430G2452 are also presented.

Software collateral and Example Project Files for Code Composer Studio™ IDE and IAR Embedded Workbench™ IDE can be downloaded from the [Capacitive Touch Software Library](#) page.

### **Contents**

1	Overview of Capacitive Touch Sensing Methods.....	2
2	Example Projects .....	5
3	References .....	18
Appendix A	Current Measurements .....	19

### **List of Figures**

1	Library Architecture.....	2
2	Capacitive Touch BoosterPack: Assignment of Sensor Elements to MCU Ports .....	5
3	Code Composer Studio New Project Wizard – Target MCU Device Selection Step.....	12
4	Code Composer Studio Project Properties Window – Predefined Preprocessor Symbols.....	13
5	Code Composer Studio Project Properties Window – Enable GCC Extensions Option.....	13
6	Code Composer Studio Project Explorer View (C/C++ Tab) .....	13
7	Code Composer Studio Project Properties Window – Include Options .....	14
8	IAR Project Options – Target Device .....	14
9	IAR Project Options – Preprocessor Options.....	15
10	IAR Project Options – FET Debugger .....	15
11	IAR Project Explorer View .....	16
12	TI Resource and MSP430ware Explorer View .....	16
13	MSP430ware Explorer View .....	17
14	Capacitive Touch Development Resources .....	18

### **List of Tables**

1	Overview of Capacitive Touch Measurement Methods That are Supported by the Library .....	3
2	Description of the Example Projects.....	11
3	Current Measurements for Example Projects .....	19

## 1 Overview of Capacitive Touch Sensing Methods

Several methods of performing capacitive touch sensing are supported when using the MSP430™ family of devices. A combination of different peripheral sets in the different device families can be used to measure the capacitive touch response from a sensor.

Figure 1 describes the architecture of the Capacitive Touch Software Library. The top layer is the Application Layer that has user-defined application code. The layers beneath the Application Layer belong to the Capacitive Touch Software Library.

The bottom layer is the Hardware Peripheral Layer that contains low-level functions for peripheral configurations. The Hardware Abstraction Layer (HAL) has functions for implementing sensing techniques using different peripheral sets.

The next layer up is the Capacitive Touch Layer that contains high-level abstract functions for implementing sensor structures such as buttons, wheels and sliders.

Note that the level of abstraction increases from left to right, where the Raw layer simply outputs raw counts as compared to the Wheel layer that outputs touch position.

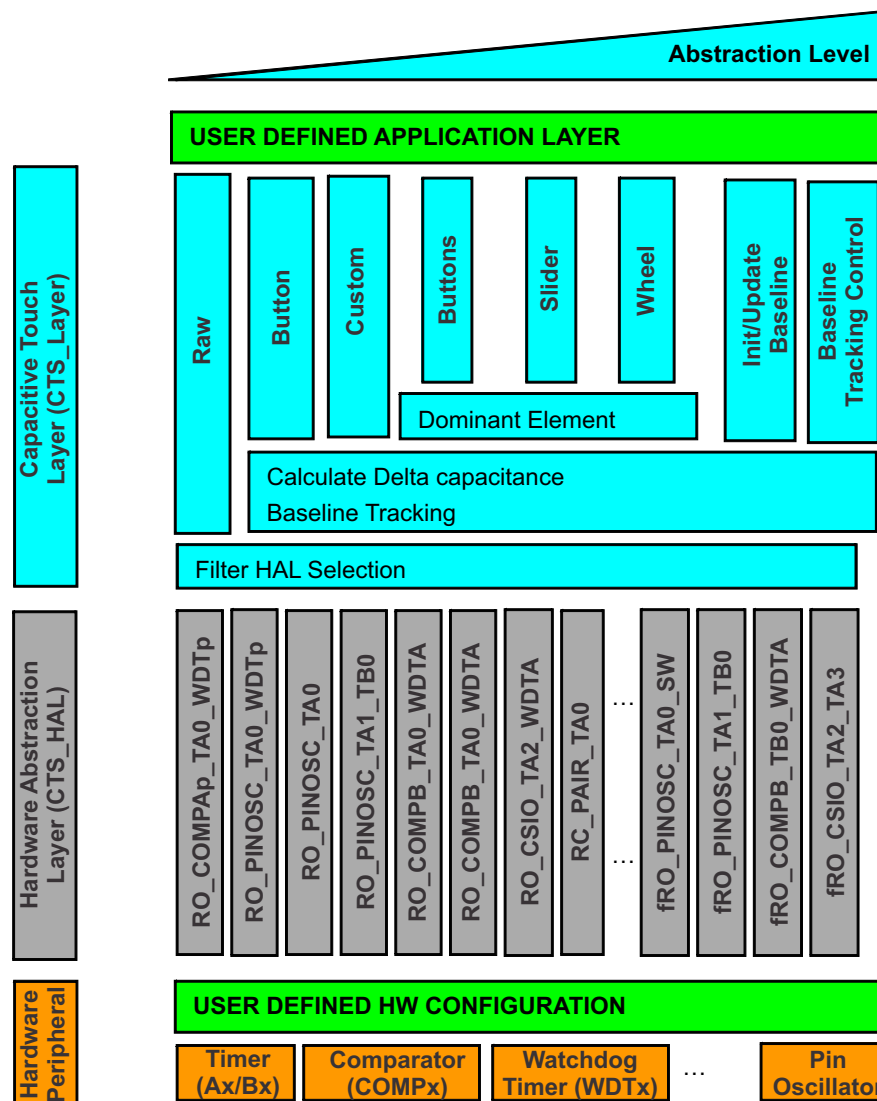


Figure 1. Library Architecture

MSP430, BoosterPack, LaunchPad, Code Composer Studio are trademarks of Texas Instruments.  
IAR Embedded Workbench is a trademark of IAR Systems AB.  
All other trademarks are the property of their respective owners.

**Table 1** lists the methods that are available in the library. The *Method Name* column shows the HAL Description Name as implemented in the Capacitive Touch Software Library. For details on each method and its implementation, see the *Capacitive Touch Software Library Programmer's Guide* (SLAA490) [1].

**Table 1. Overview of Capacitive Touch Measurement Methods That are Supported by the Library**

Method Name (HAL Description)	Sensing Type	Peripheral	Measurement Timer	Gate Timer	Benefits	Example MSP430 Devices
RO_COMPAP_TA0_WDTP	Relaxation Oscillator (RO)	Comparator COMPA+	TimerA0	Watchdog Timer WDT+	Enables low-power mode during measurements	MSP430F20x1 MSP430F4xx
RO_PINOSC_TA0_WDTP	Relaxation Oscillator (RO)	Pin Oscillator	TimerA0	Watchdog Timer WDT+	Enables low-power modes during measurements	MSP430G2xx2 MSP430G2xx3 MSP430G2xx5
RO_PINOSC_TA0	Relaxation Oscillator (RO)	Pin Oscillator	TimerA0	Software counter	Does not use watchdog timer resource	MSP430G2xx2 MSP430G2xx3 MSP430G2xx5
RO_COMPB_TA0_WDTA	Relaxation Oscillator (RO)	Comparator COMPB	TimerA0	Watchdog Timer WDTA	Enables low-power modes during measurements	MSP430F52xx MSP430F55xx CC430Fxx
RO_COMPB_TA1_WDTA	Relaxation Oscillator (RO)	Comparator COMPB	TimerA1	Watchdog Timer WDTA	Enables low-power modes during measurements	MSP430F52xx MSP430F55xx
RO_COMPB_TB0_WDTA	Relaxation Oscillator (RO)	Comparator COMPB	TimerB0	Watchdog Timer WDTA	Enables low-power modes during measurements	MSP430F52xx MSP430F55xx CC430Fxx
RO_COMPB_TA1_TA0	Relaxation Oscillator (RO)	Comparator COMPB	TimerA1	TimerA0	Enables low-power modes during measurements	MSP430F52xx MSP430F55xx
fRO_COMPB_TA1_TA0	Fast Relaxation Oscillator (fRO)	Comparator COMPB	TimerA0	TimerA1	Fast scan rate	MSP430F52xx MSP430F55xx
fRO_COMPB_TA0_SW	Fast Relaxation Oscillator (fRO)	Comparator COMPB	Software counter	TimerA0	Fast scan rate	MSP430F52xx MSP430F55xx CC430Fxx
RO_COMPB_TB0_WDTA	Relaxation Oscillator (RO)	Comparator COMPB	TimerB0	Watchdog Timer WDTA	Enables low-power modes during measurements	MSP430F52xx MSP430F55xx CC430Fxx
RC_PAIR_TA0	RC Time Constant (RC)	Digital I/O and resistor	TimerA0	N/A	Simple implementation	All MSP430 devices
fRO_PINOSC_TA0_SW	Fast Relaxation Oscillator (fRO)	Pin Oscillator	Software counter	TimerA0	Fast scan rate	MSP430G2xx2 MSP430G2xx3 MSP430G2xx5
RO_PINOSC_TA0_TA1	Relaxation Oscillator (RO)	Pin Oscillator	TimerA0	TimerA1	Enables low-power modes during measurements	MSP430G2xx3 MSP430G2xx5
fRO_PINOSC_TA0_TA1	Fast Relaxation Oscillator (fRO)	Pin Oscillator	TimerA1	TimerA0	Fast Scan Rate	MSP430G2xx3 MSP430G2xx5
RO_PINOSC_TA1_WDTP	Relaxation Oscillator (RO)	Pin Oscillator	TimerA1	Watchdog Timer WDT+	Enables low-power modes during measurements	MSP430G2xx3 MSP430G2xx5
RO_PINOSC_TA1_TB0	Relaxation Oscillator (RO)	Pin Oscillator	TimerA1	TimerB0	Enables low-power modes during measurements	MSP430G2xx5
fRO_PINOSC_TA1_TA0	Fast Relaxation Oscillator (fRO)	Pin Oscillator	TimerA0	TimerA1	Fast scan rate	MSP430G2xx3 MSP430G2xx5
fRO_PINOSC_TA1_TB0	Fast Relaxation Oscillator (fRO)	Pin Oscillator	TimerB0	TimerA1	Fast scan rate	MSP430G2xx5
RO_CSIO_TA2_WDTA	Relaxation Oscillator (RO)	Cap Touch I/O	TimerA2	Watchdog Timer WDTA	Enables low-power modes during measurements	MSP430FR58xx MSP430FR59xx
RO_CSIO_TA2_TA3	Relaxation Oscillator (RO)	Cap Touch I/O	TimerA2	TimerA3	Enables low-power modes during measurements	MSP430FR58xx MSP430FR59xx
fRO_CSIO_TA2_TA3	Fast Relaxation Oscillator (fRO)	Cap Touch I/O	TimerA3	TimerA2	Fast Scan Rate	MSP430FR58xx MSP430FR59xx
RO_CSIO_TA0_TA1	Relaxation Oscillator	Cap Touch I/O	TimerA0	TimerA1	Enables Low-Power Modes during measurements	MSP430FR4xx

**Table 1. Overview of Capacitive Touch Measurement Methods That are Supported by the Library (continued)**

Method Name (HAL Description)	Sensing Type	Peripheral	Measurement Timer	Gate Timer	Benefits	Example MSP430 Devices
RO_CSIO_TA0_WDTA	Relaxation Oscillator	Cap Touch I/O	TimerA0	Watchdog Timer WDTA	Enables Low-Power Modes during measurements	MSP430FR4xx
RO_CSIO_TA0_RTC	Relaxation Oscillator	Cap Touch I/O	TimerA0	Real-Time Clock Timer (RTC)	Enables Low-Power Modes during measurements	MSP430FR4xx
fRO_CSIO_TA0_TA1	Fast Relaxation Oscillator	Cap Touch I/O	TimerA1	TimerA0	Fast Scan Rate	MSP430FR4xx
fRO_CSIO_TA0_SW	Fast Relaxation Oscillator	Cap Touch I/O	Software Counter	TimerA0	Fast Scan Rate	MSP430FR4xx

For a description of the RO and RC principles, see *PCB-Based Capacitive Touch Sensing With MSP430* (SLAA363) [2]. For a detailed design guide for the RC-based single touch sensor, see *MSP430 Capacitive Single-Touch Sensor Design Guide* (SLAA379) [3].

## 2 Example Projects

### 2.1 Overview of the Example Projects

The example projects are built using the Capacitive Touch BoosterPack on the following development kits:

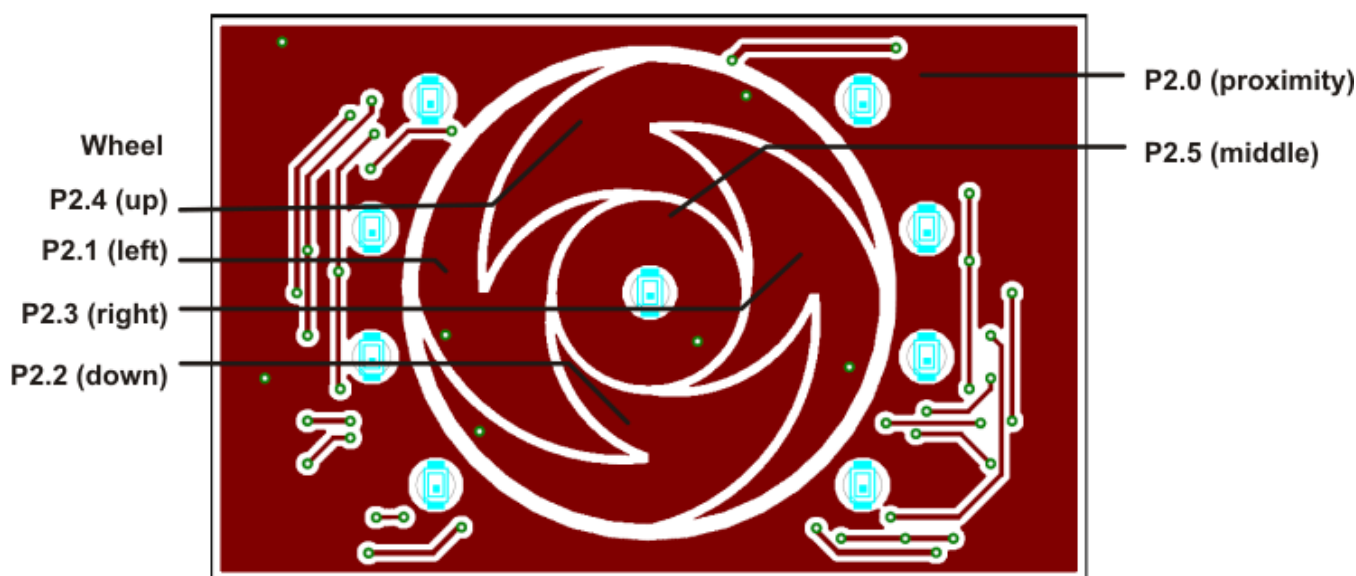
- MSP430 LaunchPad Value Line Development Kit ([MSP-EXP430G2](#)) with the MSP430G2452 MCU
- MSP430FR5969 LaunchPad Development Kit ([MSP-EXP430FR5969](#)) with the MSP430FR5969 MCU
- MSP430FR4133 LaunchPad Development Kit ([MSP-EXP430FR4133](#)) with the MSP430FR4133 MCU

This document describes the projects by using the Value Line LaunchPad development kit with the MSP430G2452 MCU as examples.

The Value Line LaunchPad is a low-cost development kit with a 20-pin PDIP socket, on-board programmer and debugger, universal serial bus (USB) communication to PC, and other features [4]. Along with the 8 KB of Flash and 256 B of RAM, the MSP430G2452 MCU has 16 capacitive-touch enabled input/output (I/O) pins that use the pin oscillator feature mentioned in [Section 1](#).

### 2.2 Overview of the Capacitive Touch BoosterPack Hardware

The Capacitive Touch BoosterPack is based on the pin oscillator feature and includes three types of sensors: Proximity, Button, and Wheel (group of buttons). [Figure 2](#) shows the layout of the sensors and the port connections on the MSP430G2452 MCU.



**Figure 2. Capacitive Touch BoosterPack: Assignment of Sensor Elements to MCU Ports**

There are also nine LED elements that are multiplexed to indicate touch: four on either side and one for the middle element. For details on the schematic connections and PCB layout, see the *Capacitive Touch BoosterPack User's Guide* ([SLAU337](#)) [6].

## 2.3 Configuring the Software Library

The Capacitive Touch Software Library must be configured with the port definition, sensing method, number of elements, and other factors. These factors can be configured in `structure.c` source code and `structure.h` header files. The following steps are presented using the `RO_PINOSC_TA0_WDTp_One_Button` Example where the sensing method is `RO_PINOSC_TA0_WDTp` and the sensor structure comprises of one button (the middle element).

1. **Element Definition:** Declare the element structure and assign the port definition.

Example: The middle element is mapped to P2.5 (Port 2 Pin 5).

Within the `structure.c` source code file:

```
// Middle Element (P2.5)
const struct Element middle_element = {

    .inputPxselRegister = (uint8_t *) &P2SEL,
    .inputPxsel2Register = (uint8_t *) &P2SEL2,
    .inputBits = BIT5,
    // When using an abstracted function to measure the element
    // the 100*(maxResponse - threshold) < 0xFFFF
    // ie maxResponse - threshold < 655
    .maxResponse = 200+655,
    .threshold = 200    // Set threshold to zero for element characterization
};
```

Within the `structure.h` header file:

```
extern const struct Element middle_element;
```

---

**NOTE:** The `threshold` variable should be set to zero for the element characterization step.

---



---

**NOTE:** The `maxResponse` variable is required only for the wheel or slider implementation and is not needed for a single sensor implementation. For more details, see the *Capacitive Touch Software Library Programmer's Guide* (SLAA490) [1].

---

2. **Sensor Structure Definition:** Declare the sensor structure and define the sensing method, number of sensor elements, the gate timer source, and the measurement window.

Example:

The sensor structure:

- Composed of one element (the middle element)
- Sensing method is the relaxation oscillator using the pin oscillator feature
- TimerA0 is measurement timer; watchdog timer (WDT) is gate timer

Within the `structure.c` source code file:

```
// One Button Sensor
const struct Sensor one_button =
{
    .halDefinition = RO_PINOSC_TA0_WDTp, // Sensing Method
    .numElements = 1, // # of Elements
    .baseOffset = 0, // First element index = 0
    // Pointer to elements
    .arrayPtr[0] = &middle_element, // point to middle element
    // Timer Information
    .measGateSource= GATE_WDT_ACLK, // 0→SMCLK, 1→ ACLK
    .accumulationCycles= WDTp_GATE_64 // 64 - Default
};
```

Within the `structure.h` header file:

```
extern const struct Sensor one_button;
```

### 3. Characterizing Element Performance:

**Initialize Threshold:** As mentioned in Step 1, the threshold value should be initialized to zero for element characterization.

**Configure the Gate Timer:** This example uses the `RO_PINOSC_TA0_WDTp` method with the Watchdog timer used as the Gate Timer. The Gate Time has a direct impact on the noise immunity, touch sensitivity, power consumption, and overall performance of the sensor.

- Adjust the Gate Timer Source (`measGateSource`) to source the WDT from either low-frequency ACLK or high-frequency SMCLK.
- Adjust the measurement window (`accumulationCycles`) to select between different WDT intervals.

Within the `structure.c` source code file:

```
// One Button Sensor
const struct Sensor one_button =
{
    .halDefinition = RO_PINOSC_TA0_WDTp, // Sensing Method
    .numElements = 1, // # of Elements
    .baseOffset = 0, // First element index = 0
    // Pointer to elements
    .arrayPtr[0] = &middle_element, // point to middle element

    // Timer Information
    // Select Gate Timer Source for Watchdog Timer
    // .measGateSource= GATE_WDT_SMCLK, // SMCLK?
    .measGateSource= GATE_WDT_ACLK, // ACLK?

    // Select Measurement Window time frame by selecting
    // the Watchdog Timer Input Clock divider setting
    // .accumulationCycles= WDTp_GATE_32768 // 32768
    // .accumulationCycles= WDTp_GATE_8192 // 8192
    // .accumulationCycles= WDTp_GATE_512 // 512
    .accumulationCycles= WDTp_GATE_64 // 64
};
```

**Set Up the Characterization Function:** After the Gate Time has been configured, the `main()` function should call the Baseline tracking function and monitor the counts returned from the `TI_CAPT_Custom()` function. This action can be accomplished by un-commenting or declaring the following compiler

directive in the `main.c` source code file.

```
// Uncomment to have this compiler directive run characterization functions only
#define ELEMENT_CHARACTERIZATION_MODE
```

When the compiler directive `ELEMENT_CHARACTERIZATION_MODE` is defined, the following code for element characterization is compiled within `main.c` source code file.

```
// Within the application layer source code file: main.c
#include "CTS_Layer.h"

// Delta Counts returned from the API function for the sensor during characterization
unsigned int dCnt;

void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;          // Stop watchdog timer
    .....// Set up system clocks
    BCSCTL3 |= LFXT1S_2;                // LFXT1 = VLO Clock Source

    P1OUT = 0x00;                       // Drive all Port 1 pins low
    .....// Initialize Ports
    P2DIR = 0xFF;                       // Configure all Port 2 pins outputs

    // Initialize Baseline measurement
    TI_CAPT_Init_Baseline (&one_button,5);

    // Update baseline measurement (Average 5 measurements)
    TI_CAPT_Update_Baseline (&one_button,5);

    while (1)
    {
        // Get the raw delta counts for element characterization
        TI_CAPT_Custom(&one_button,&dCnt);
        __no_operation();               // Set breakpoint here
    }
}
```

**Get the Characterization Delta Counts:** To get the delta counts for element characterization, set a breakpoint on the NOP following the `TI_CAPT_Custom()` function and add the variable `dCnt` to the Watch window. Run the program several times and record the value for the following two cases:

- **No Touch:** No finger is touching the middle element and no conductive object in the immediate vicinity of the sensor. This is effectively characterizing the background noise. Record the maximum value of `dCnt` in this scenario as *noTouchCnt*.
- **Touch:** A finger is touching the middle element or some conductive object is in the immediate vicinity of the sensor. This is effectively characterizing the touch response sensitivity. Record the minimum value of `dCnt` in this scenario as *TouchCnt*.



4. **Set the Threshold Value:** The threshold value should nominally be set to the middle of the range:  $threshold = (TouchCnt - noTouchCnt)/2$ . The threshold can be set higher or lower to increase noise immunity or increase sensitivity.

---

**NOTE:** Setting a threshold value too low increases the probability of false triggers (and make it susceptible to noise). Setting a threshold value too high causes the sensor to not recognize the presence of a conductive object.

---

After the threshold value is calculated, the respective element structure should be updated with the new value. For example, during the characterization process,  $noTouchCnt = 100$  and  $TouchCnt = 1000$ . The *threshold* value should be set to 450.

Within `structure.c` source code file:

```
// Middle Element (P2.5)
const struct Element middle_element = {

    .inputPxselRegister = (uint8_t *) &P2SEL,
    .inputPxsel2Register = (uint8_t *) &P2SEL2,
    .inputBits = BIT5,
    // When using an abstracted function to measure the element
    // the 100* (maxResponse - threshold) < 655
    // ie maxResponse - threshold < 655
    .maxResponse = 450+655,
    .threshold = 450      // Update the threshold after characterization
}
```

---

**NOTE:** Steps 3 and 4 may have to be attempted multiple times to determine the appropriate gate time for the PCB layout, neighboring element structures, and application noise levels. This is an iterative fine-tuning process.

---

5. **Use API Calls to Achieve Higher Levels of Abstraction:** The library provides several API calls that can abstract all of the inner workings, such as baseline tracking and rate of change. These calls provide the advantage of having streamlined code in the Application Layer.

---

**NOTE:** All of the measurement API functions (with the exception of the `TI_CAPT_RAW()` function) update the baseline tracking. The `TI_CAPT_Update_Baseline()` and `TI_CAPT_Init_Baseline()` functions should be used if the sensor is not measured for a long period of time (because drift in supply voltage, temperature, or environmental conditions may occur during this time).

---

**Objective of the Sample Application:** To detect touch of one button (middle element), the application calls the `TI_CAPT_Button()` API function. The function returns logic one or zero to indicate if there is a valid touch or not. This information is used to turn on the center LED on the Capacitive Touch BoosterPack. The LED serves as a visual indicator of center button touch.

Between consecutive polling calls, the MSP430 MCU is placed into low-power mode (LPM3). Timer A0 is used to implement a delay timeout feature to wake up from LPM3 into Active Mode. This delay is programmable and can be adjusted in `main.c` source code file.

```
#define DELAY 5000                // Timer delay timeout count - 5000*0.1msec = 500 ms
```

**Configure the Sample Application:** In `main.c` source code file, the compiler directive `ELEMENT_CHARACTERIZATION_MODE` should be commented out so that the application code (rather than characterization code) is compiled.

```
// Comment to have this compiler directive run example application
// #define ELEMENT_CHARACTERIZATION_MODE
```

## 2.4 Example Projects

[Example 1](#) shows the directory and file structure for the associated files that are available for download with this application report.

### Example 1. File and Directory Structure

```

+--- Code Examples
+--- Documentation
+--- Library
    +--- CTS_HAL.c
    +--- CTS_HAL.h
    +--- CTS_Layer.c
    +--- CTS_Layer.h
+--- Examples_projects
+--- CCS
+--- IAR
+--- Source
    +--- RO_PINOSC_TA0_WDTp_One_Button
        main.c
        structure.c
        structure.h
    +--- RO_PINOSC_TA0_WDTp_One_Button_Compact
        main.c
        structure.c
        structure.h
    +--- RO_PINOSC_TA0_WDTp_Proximity_Sensor
        main.c
        structure.c
        structure.h
    +--- RO_PINOSC_TA0_WDTp_Wheel_Buttons
        main.c
        structure.c
        structure.h
    +--- CSIO_TA2_WDTA_One_Button
        main.c
        structure.c
        structure.h
    +--- CSIO_TA2_WDTA_Proximity_Sensor
        main.c
        structure.c
        structure.h
    +--- CSIO_TA2_WDTA_Wheel_Buttons
        main.c
        structure.c
        structure.h

```

- **CCS:** Contains Project folders and files for Code Composer Studio IDE.
- **IAR:** Contains Project folders and files for IAR Embedded Workbench.
- **Library:** Contains the Capacitive Touch Library Files (HAL and Application Layer).
- **Source:** Each folder contains source code for different sensor methods (sample application and structure definition files) based on different MSP430 devices.

The high-level API calls such as *TI\_CAPT\_Custom()*, *TI\_CAPT\_Buttons()* and so forth consume more Flash memory space but abstract the inner workings such as Baseline Tracking, Rate of Updates, and Direction of Interest from the application layer. The resulting Application Layer is compact and any updates to the HAL or Layer functions do not require any modifications to the application code.

The low-level API calls such as *TI\_CAPT\_Raw()* can be used to achieve lower Flash memory code footprint, but features such as baseline tracking must be implemented manually in the application code. This scenario can possess an advantage for devices with little memory and yet obtain very low levels of power consumption.

**Table 2. Description of the Example Projects**

Example Name	API Level	Sensor Structure	Description	Flash <sup>(1)(2)</sup> (bytes)	RAM (bytes)
RO_PINOSC_TA0_WDTp_One_Button_Compact	Low	Button (Middle Element)	Turn the center LED on or off based on middle element touch	718	6
RO_PINOSC_TA0_WDTp_One_Button	High	Button (Middle Element)	Turn the center LED on or off based on middle element touch	2330	8
RO_PINOSC_TA0_WDTp_Proximity_Sensor	High	Proximity Sensor	Turn the center LED on or off based on proximity sensor detection	1910	14
RO_PINOSC_TA0_WDTp_Wheel_Buttons	High	Wheel (Group of Five Buttons)	Turn the center LED on or off based on the five wheel buttons	2390	16
RO_CSIO_TA2_WDTA_One_Button_Compact	Low	Button (Middle Element)	Turn the center LED on or off based on middle element touch	688	6
RO_CSIO_TA2_WDTA_One_Button	High	Button (Middle Element)	Turn the center LED on or off based on middle element touch	2144	6
RO_CSIO_TA2_WDTA_Proximity_Sensor	High	Proximity Sensor	Turn the center LED on or off based on proximity sensor detection	2144	6
RO_CSIO_TA2_WDTA_Wheel_Buttons	High	Wheel (Group of Five Buttons)	Turn the center LED on or off based on proximity sensor detection	2520	24
RO_CSIO_TA0_RTC_One_Button_Compact	Low	Button (Middle Element)	Turn the center LED on or off based on middle element touch	718	6
RO_CSIO_TA0_RTC_One_Button	High	Button (Middle Element)	Turn the center LED on or off based on middle element touch	2174	6
RO_CSIO_TA0_RTC_Proximity_Sensor	High	Proximity Sensor	Turn the center LED on or off based on proximity sensor detection	2174	6
RO_CSIO_TA0_RTC_Wheel_Buttons	High	Wheel (Group of Five Buttons)	Turn the center LED on or off based on the five wheel buttons	2542	24

<sup>(1)</sup> CCS 6.0.0 IDE was used to generate the Flash and RAM memory allocation results and the optimization level is 0 Register Optimizations.

<sup>(2)</sup> The stack size (80 bytes in this example) is not included in the RAM size.

**NOTE:** The above results are based on the compile results using Code Composer Studio v6.0.0 with MSP430 Code Generation Tools v4.3.1 with the device of MSP430G2452(PINOSC), MSP430FR5969(CSIO) and MSP430FR413.

## 2.5 Project Setup in Code Composer Studio IDE

The Code Composer Studio (CCS) Integrated Development Environment (IDE) can be used to build, download, and debug the project source code.

**NOTE:** Code Composer Studio 5.4.0 with MSP430 Code Generation Tools v5.3.0 was used to create the example projects.

1. Initiate Code Composer Studio, File → New CCS Project, and follow the steps through the New Project Wizard to create a project for an example. For more information on new project setup, see the *Code Composer Studio v5.3 User's Guide for MSP430 User's Guide* (SLAU157) [7].
2. Select the CCS project location and the appropriate target MCU device. In the case of the Capacitive Touch BoosterPack, select the MSP430G2452 MCU (as shown in Figure 3).

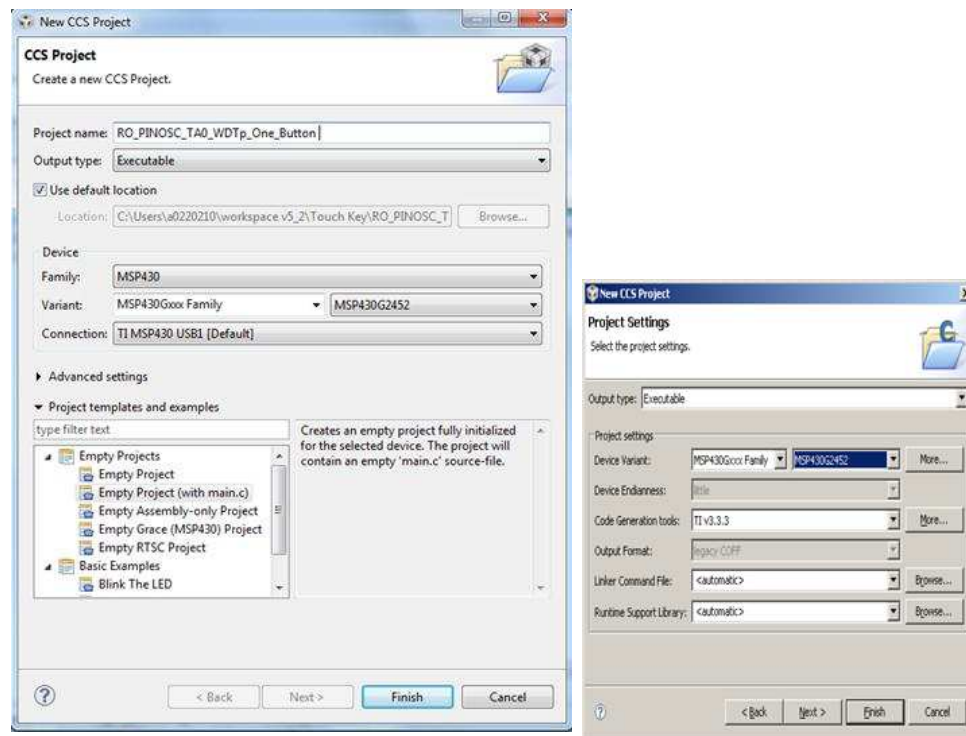
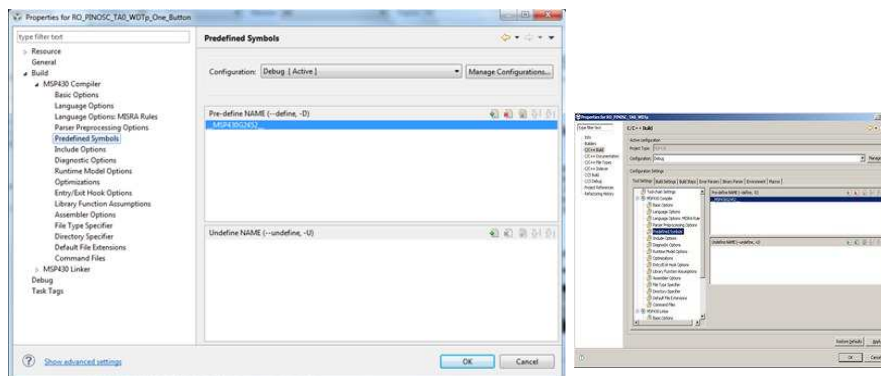


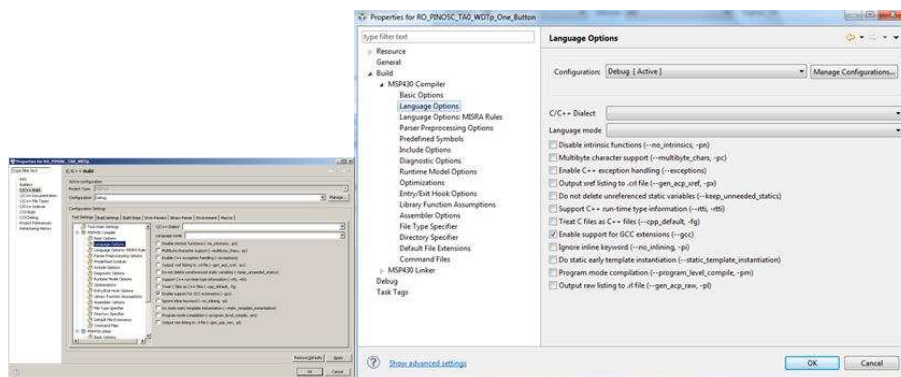
Figure 3. Code Composer Studio New Project Wizard – Target MCU Device Selection Step

3. Ensure that the preprocessor and predefined symbol in the Project Build Options matches the target MCU device (as displayed in [Figure 4](#)).



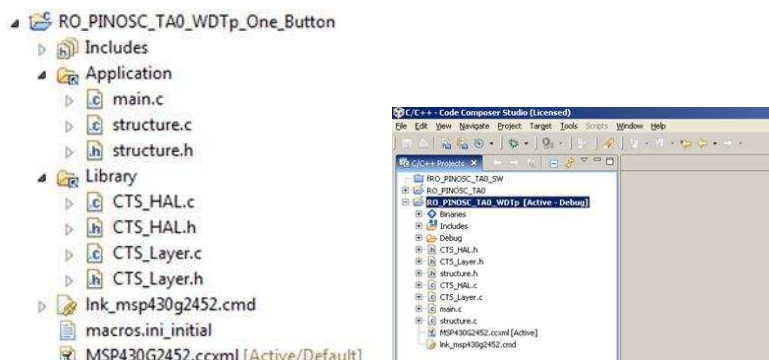
**Figure 4. Code Composer Studio Project Properties Window – Predefined Preprocessor Symbols**

4. Enable the GCC Extensions for the Project Build (as shown in [Figure 5](#)).
  - Go to Project → Properties to bring up the Project properties dialog.
  - On left side, select Build → MSP430 Compiler → Advanced Options → Language Options
  - On right side, under Configuration Settings, select the check box of Enable Support for GCC extensions (--gcc).



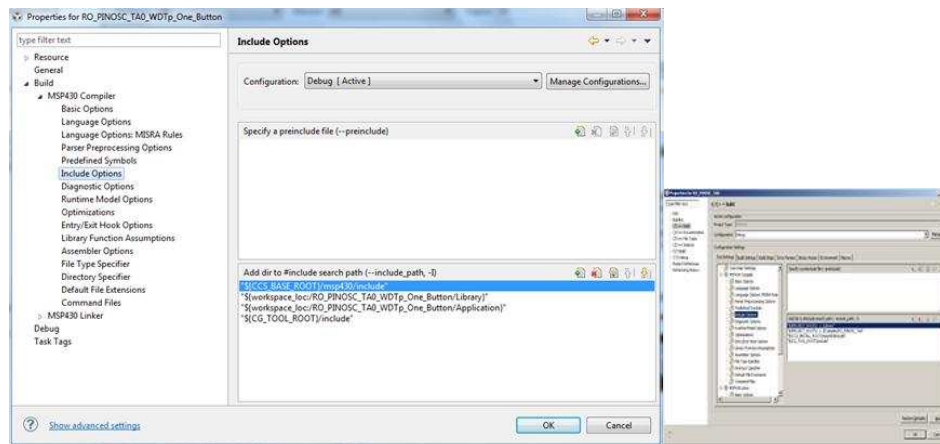
**Figure 5. Code Composer Studio Project Properties Window – Enable GCC Extensions Option**

5. Link the Capacitive Touch Software Library (HAL and Layer) and the Example Project files (Main and Structure) to the project. The Project configuration should look similar to [Figure 6](#).



**Figure 6. Code Composer Studio Project Explorer View (C/C++ Tab)**

6. Add the path of the directories to the Include Path Option (as shown in [Figure 7](#)).



**Figure 7. Code Composer Studio Project Properties Window – Include Options**

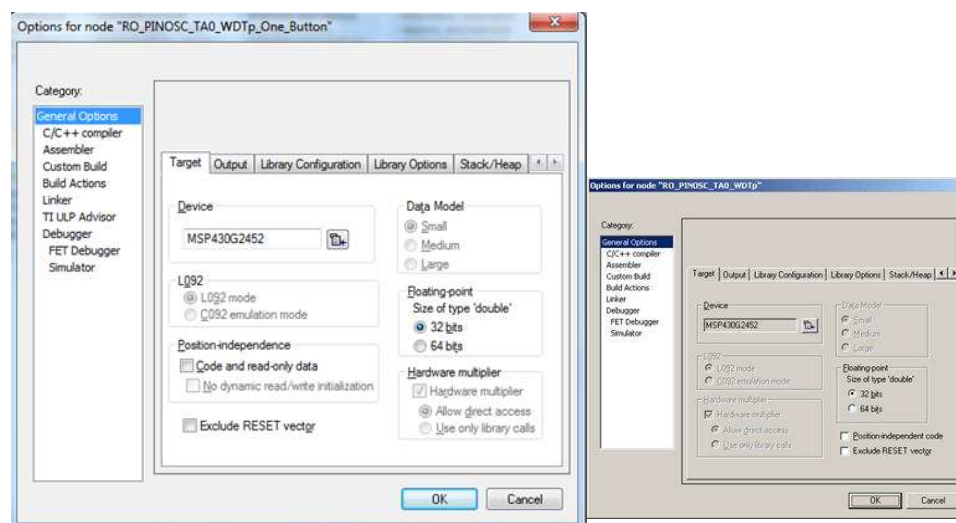
7. Download and debug the example project: Target → Debug Active Project.

## 2.6 Project Setup in IAR Embedded Workbench IDE

The IAR Embedded Workbench (IAR) Integrated Development Environment (IDE) can be used to build, download, and debug the project source code.

**NOTE:** IAR Embedded Workbench 6.4 with MSP430 v5.50 was used to create the example projects.

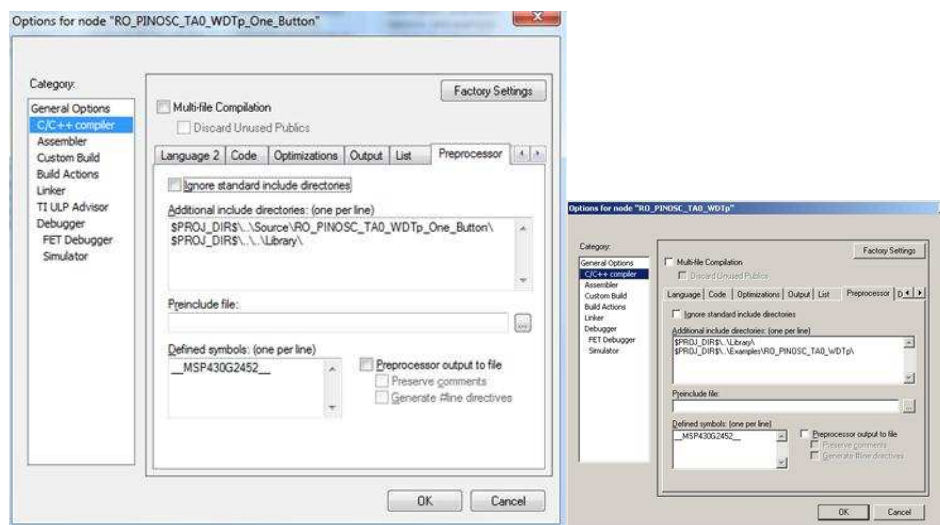
1. Initiate IAR, Project → Create New Project, and follow through the steps of giving a new project name with MSP430 to create a new project. For more information on new project setup, see the *IAR Embedded Workbench Version 3+ for MSP430 User's Guide* (SLAU138) [8].
2. To select the appropriate target device, click Project → Options, then click General Options and the Target Tab. In the case of the Capacitive Touch BoosterPack, select the MSP430G2452 (as shown in [Figure 8](#)).



**Figure 8. IAR Project Options – Target Device**

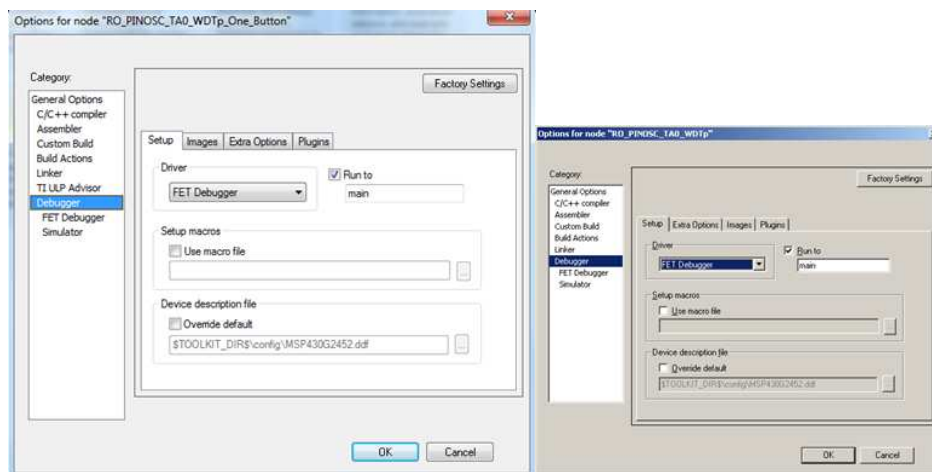


3. Ensure that the preprocessor and predefined symbol in the Project Options → C/C++ Compiler matches the target MCU device (as displayed in Figure 9). Add the path of the directories to the Additional include directories input box.



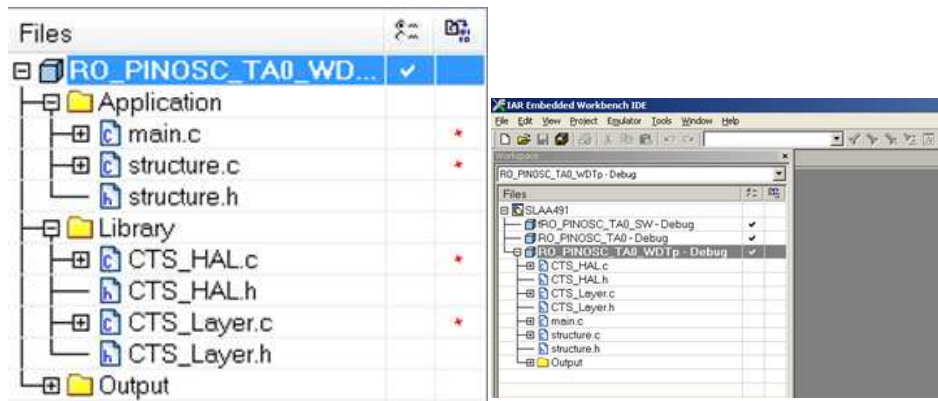
**Figure 9. IAR Project Options – Preprocessor Options**

4. Select FET Debugger as the Debugger option to download and debug the example project (as shown in Figure 10).



**Figure 10. IAR Project Options – FET Debugger**

5. Add the Capacitive Touch Software Library (HAL and Layer) and the Example Project files (Main and Structure) to the project: Project → Add Files. The Project configuration should look similar to Figure 11.



**Figure 11. IAR Project Explorer View**

6. Download and debug the example project: Project → Download and Debug.

## 2.7 Project Setup Using MSP430ware

MSP430ware is a collection of code examples, data sheets, and other design resources for all MSP430 devices delivered in a convenient package, it is available either as a component of Code Composer Studio (CCS) or as a standalone package. A capacitive touch project is easy to build using MSP430ware.

**NOTE:** Code Composer Studio v5.3.0 with MSP430 Code Generation Tools v3.3.3 was used to create the example projects.

1. Start Code Composer Studio and find the MSP430ware menu. Click View → TI Resource Explorer to open the interface shown in Figure 12.



**Figure 12. TI Resource and MSP430ware Explorer View**

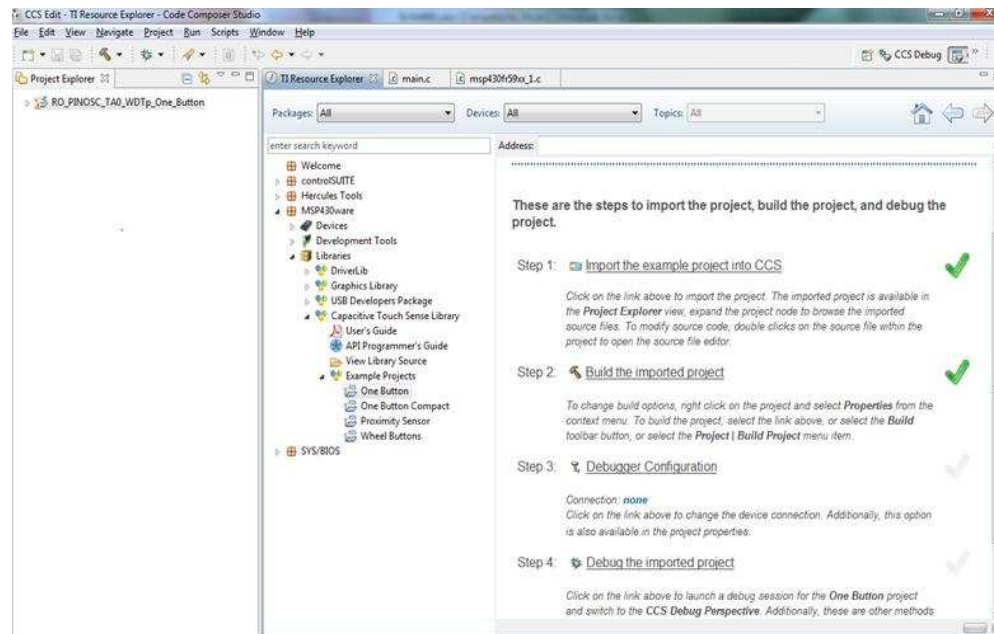


2. Select MSP430ware → Libraries → Capacitive Touch Software Library → Example Projects. Several example projects of capacitive touch, several example projects of capacitive touch, like one button, wheel buttons will be listed, as shown in [Figure 13](#).



**Figure 13. MSP430ware Explorer View**

3. Click One Button in the list of Example Projects, several steps to import, build, and debug the project will be illustrated on the right side. The following different steps, different operations will be applied to the selected Project. For example, if click Step 1: Import the example project into CCS, a project named RO\_PINOSC\_TA0\_WDTp\_One\_Button will turn up on the left Project explorer, as shown in Figure 14.



**Figure 14. Capacitive Touch Development Resources**

### 3 References

1. *Capacitive Touch Software Library Programmer's Guide* ([SLAA490](#))
2. *PCB-Based Capacitive Touch Sensing With MSP430* ([SLAA363](#))
3. *MSP430 Capacitive Single-Touch Sensor Design Guide* ([SLAA379](#))
4. *MSP-EXP430G2 LaunchPad Experimenter Board User's Guide* ([SLAU318](#))
5. *MSP430G2x52, MSP430Gx12 Mixed Signal Microcontroller Data Sheet* ([SLAS722](#))
6. *Capacitive Touch BoosterPack (430BOOST-SENSE1) for the LaunchPad User's Guide* ([SLAU337](#))
7. *Code Composer Studio v5.3 for MSP430 User's Guide* ([SLAU157](#))
8. *IAR Embedded Workbench Version 3+ for MSP430 User's Guide* ([SLAU138](#))
9. *Capacitive Touch Hardware Design Guide* ([SLAA576](#))

## Current Measurements

Table 3 shows the current measurements for the example projects. Note that these measurements represent typical (averaged) values.

**Table 3. Current Measurements for Example Projects**

Example Name	Configuration	Voltage	Current	Range
RO_PINOSC_TA0_WDTP_One_Button_Compact	One Middle Element Button	3.3 V	2.4 $\mu$ A	Touch
	Scan Time = 500 ms	3.0 V	2.0 $\mu$ A	
	WDT source = ACLK/64	2.5 V	1.4 $\mu$ A	
	ACLK = VLO ( $\approx$ 12 kHz)	1.8 V	0.9 $\mu$ A	
	Gate Time $\approx$ 5.3 ms per element			
RO_PINOSC_TA0_WDTP_One_Button	One Middle Element Button	3.3 V	2.6 $\mu$ A	Touch
	Scan Time = 500 ms	3.0 V	2.1 $\mu$ A	
	WDT source = ACLK/64	2.5 V	1.7 $\mu$ A	
	ACLK = VLO ( $\approx$ 12 kHz)	1.8 V	1 $\mu$ A	
	Gate Time $\approx$ 5.3 ms per element			
RO_PINOSC_TA0_WDTP_Proximity_Sensor	STRUCTURE_CONFIG_0 <sup>(1)</sup>	3.3 V	2.5 $\mu$ A	1.5 cm
	Scan Time = 500 ms	3.0 V	2 $\mu$ A	1 cm
	WDT source = ACLK/64	2.5 V	1.5 $\mu$ A	0.5 cm
	ACLK = VLO ( $\approx$ 12 kHz)	1.8 V	1.1 $\mu$ A	Touch
	Gate Time $\approx$ 5.3 ms per element			
RO_PINOSC_TA0_WDTP_Proximity_Sensor	STRUCTURE_CONFIG_1 <sup>(1)</sup>	3.3 V	9.5 $\mu$ A	2.5 cm
	Scan Time = 500 ms	3.0 V	8 $\mu$ A	2 cm
	WDT source = ACLK/512	2.5 V	6 $\mu$ A	1.8 cm
	ACLK = VLO ( $\approx$ 12 kHz)	1.8 V	3 $\mu$ A	1.5 cm
	Gate Time $\approx$ 42.6 ms per element			
RO_PINOSC_TA0_WDTP_Proximity_Sensor	STRUCTURE_CONFIG_2 <sup>(1)</sup>	3.3 V	20 $\mu$ A	3.5 cm
	Scan Time = 500 ms	3.0 V	17.5 $\mu$ A	3 cm
	WDT source = SMCLK/8192	2.5 V	14 $\mu$ A	2.8 cm
	SMCLK = DCO ( $\approx$ 1 MHz)	1.8 V	9 $\mu$ A	2.5 cm
	Gate Time $\approx$ 65.5 ms per element			
RO_PINOSC_TA0_WDTP_Wheel_Buttons	Four Wheel Buttons + One Middle Element Button	3.3 V	8.5 $\mu$ A	Touch
	Scan Time = 500 ms	3.0 V	7 $\mu$ A	
	WDT source = ACLK/64	2.5 V	4.8 $\mu$ A	
	ACLK = VLO ( $\approx$ 12 kHz)	1.8 V	2.6 $\mu$ A	
	Gate Time $\approx$ 5.3 ms per element			

<sup>(1)</sup> The compiler directive for the structure configuration should be defined at the top of `structure.c` source file.

### Configuration for Current Measurement

- On the LaunchPad, remove all Jumpers on J3: V<sub>CC</sub>, TXD, RXD, RST, and TEST.
- On the LaunchPad, remove resistor R34, the pullup resistor on P1.3 and pushbutton switch S2.

**NOTE:** The removal of R34 is required for low-current measurements.

- If a pullup resistor is required, configure the port registers on P1.3 to use the internal on-chip pullup

resistors.

## Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

### Changes from D Revision (September 2014) to E Revision Page

- 
- Added full LaunchPad names and links in [Section 2.1](#) ..... 5
-

## 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

Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>
OMAP Applications Processors	<a href="http://www.ti.com/omap">www.ti.com/omap</a>
Wireless Connectivity	<a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a>

### Applications

Automotive and Transportation	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
Communications and Telecom	<a href="http://www.ti.com/communications">www.ti.com/communications</a>
Computers and Peripherals	<a href="http://www.ti.com/computers">www.ti.com/computers</a>
Consumer Electronics	<a href="http://www.ti.com/consumer-apps">www.ti.com/consumer-apps</a>
Energy and Lighting	<a href="http://www.ti.com/energy">www.ti.com/energy</a>
Industrial	<a href="http://www.ti.com/industrial">www.ti.com/industrial</a>
Medical	<a href="http://www.ti.com/medical">www.ti.com/medical</a>
Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
Space, Avionics and Defense	<a href="http://www.ti.com/space-avionics-defense">www.ti.com/space-avionics-defense</a>
Video and Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>

### TI E2E Community

[e2e.ti.com](http://e2e.ti.com)