

ABSTRACT

Capacitive touch sensing is a technology that detects when a finger approaches or touches the touch surface through changes in capacitance. Through capacitive sensing, mechanical switches and knobs can be replaced with elegant buttons, sliders and scroll wheels.

Table of Contents

1 Introduction	2
1.1 Overview.....	2
1.2 Terms and Abbreviations.....	3
2 Principles of Capacitive Touch	3
2.1 Self-Capacitive Detection.....	3
2.2 Mutual-Capacitive Detection.....	5
2.3 TI's Capacitive Touch Technology.....	6
3 MCU Selection and Evaluation	8
3.1 Determine Sensor Requirements.....	8
3.2 MCU Selection.....	10
3.3 EVM Development Board Selection and Evaluation.....	10
4 Mechanical Structure and Hardware Design	12
4.1 Mechanical Structure Design.....	12
4.2 Hardware Design.....	14
5 Software Design and Parameter Tuning	17
5.1 Concepts Required for CapTIvate Software Development.....	17
5.2 Phase 1: GUI Configuration.....	21
5.3 Phase 2: Download Code.....	24
5.4 Phase 3: Adjust Parameters.....	24
5.5 Phase 4: Modify Communication Mode.....	28
5.6 Phase 5: Develop Custom Applications.....	28
6 Installation and Process Overview	34
6.1 Installation of Code Composer Studio IDE.....	34
6.2 Installation of CapTIvate Design Center.....	35
6.3 Hardware Connection.....	37
6.4 Rapid Evaluation.....	39
6.5 Rapid Development.....	42
7 References	49
8 Revision History	49

Trademarks

CapTIvate™, MSP430™, and Code Composer Studio™ are trademarks of Texas Instruments. All trademarks are the property of their respective owners.

1 Introduction

1.1 Overview

Physical button, knobs, and sliders experience the following shortcomings compared to capacitive touch controls. Physical controls:

1. Experience wear and decreased reliability after prolonged use.
2. Require a gap between the front panel and the buttons, which is easy to be penetrated by moisture that can cause defects.
3. Require the user to exert force to trigger.
4. Require openings in the front panel that can increase cost to a certain extent.
5. Have relative fixed shapes.

TI CapTIvate™ capacitive touch technology supports five sensor types: buttons, proximity sensors, scroll wheels, sliders, and touch panels, and a variety of covering materials. CapTIvate technology supports low power consumption, strong and stable induction technology, strong anti-noise ability, and allows for waterproof construction.

This document helps with the capacitive touch development process and helps you quickly understand the full features of TI CapTIvate capacitive touch technology.

- For a preliminary function evaluation, see [Section 2](#), [Section 3](#), and [Section 6](#).
- For hardware and mechanical structure development, see [Section 2](#), [Section 3](#), and [Section 4](#).
- For software development, see [Section 3](#), [Section 5](#), and [Section 6](#).

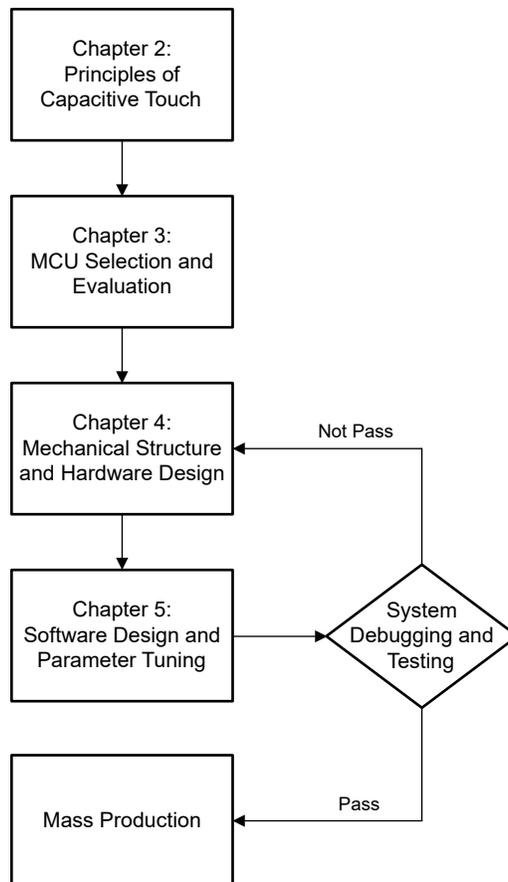


Figure 1-1. Capacitive Touch Development Process

1.2 Terms and Abbreviations

Term	Definition
Base capacitance	The parasitic capacitance of the sensor before the finger touches it
BSL	Bootloader
CAP I/O	A pin on an MSP430™ MCU that is dedicated to the capacitive touch function
CapTIvate	TI's capacitive touch design system
FRAM	Ferroelectric RAM (also FeRAM or F-RAM)
GUI	Graphical user interface
IDE	Integrated development environment
JTAG	JTAG (named after the Joint Test Action Group) is an industry standard for verifying designs, testing printed circuit boards programming and debugging after manufacture
LPM0, LPM3, LPM4	Different low-power modes of MSP430 MCUs. For specific power consumption, see the device-specific data sheet.
MSP	Mixed signal processor
NVM	Nonvolatile memory
Rx	In mutual-capacitance mode or self-capacitance mode, the pin or electrode responsible for charging from the parasitic capacitance to the internal reference capacitance of the MCU.
SBW	2-wire Spy-Bi-Wire interface, a typical JTAG interface for MSP430 MCUs
Tx	In mutual-capacitance or self-capacitance mode, the pin or electrode responsible for charging the parasitic capacitance.

2 Principles of Capacitive Touch

Figure 2-1 shows a common capacitive touch sensor. Sensors typically use copper on the PCB as electrodes. The top layer is covered with a nonconductive protective layer, such as glass or plastic, and glued to the PCB. A grid ground surrounds the sensor.

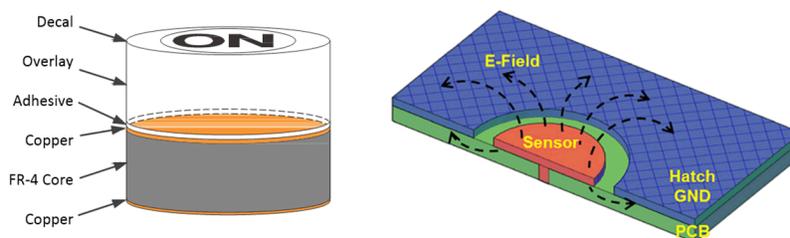


Figure 2-1. Capacitive Touch Structure

Based on the type of capacitance detected, capacitive touch can be described as self-capacitive detection (detecting the capacitance value between a single electrode and power line ground) or mutual-capacitive detection (detecting the capacitance value between two electrodes).

2.1 Self-Capacitive Detection

Taking the simplest single button as an example, Figure 2-2 shows a self-inductive capacitor, and Figure 2-3 shows the detection model. The self-inductive capacitor uses a single electrode (receiving electrode Rx) formed by copper coating to detect the change in capacitance of the electrode to the power line ground. The initial capacitance of the button to power line ground is C_p . When a human hand touches it, C_t , C_h , and C_g are introduced into the circuit, thereby **increasing** the capacitance of the button to ground.

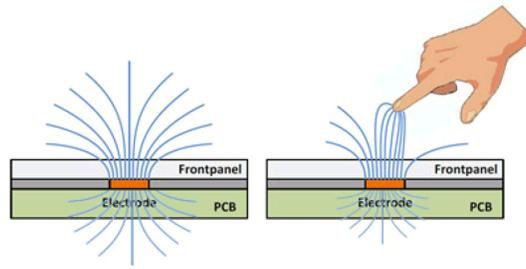


Figure 2-2. Self-Capacitive Detection

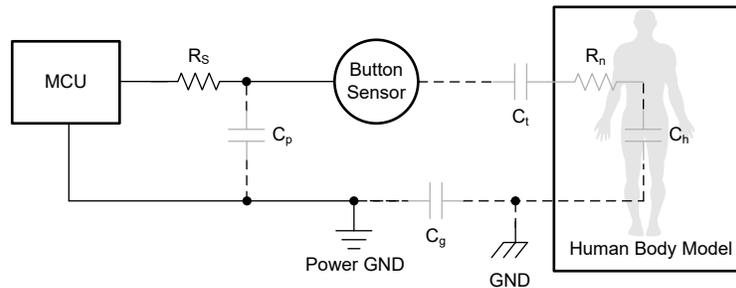


Figure 2-3. Self-Capacitive Detection Model

In [Figure 2-3](#), the solid line indicates the PCB routing, and the dashed line indicates the touch-related routing. Gray components indicate equivalent capacitance or resistance.

R_h = Human body resistance.

R_s = Series resistance. The recommend value is 470 Ω .

C_p = The parasitic capacitance of the button and the connected wire to the power line ground.

C_g = The capacitance between the power line ground and the earth ground. For battery applications, it is approximately 1pF. For grounding applications, it is a short circuit.

C_h = Series capacitance between the human body and the earth ground.

C_t = The capacitance formed by the electrical level and the human fingertip, which is similar to the structure of a parallel plate capacitance.

For ease of analysis, the influence of R_h and R_s is ignored. [Equation 1](#) shows the equivalent capacitance of the button to the power line ground. Sensitivity can be characterized as the ratio between the capacitance change caused by the touch and the base capacitance, as shown in [Equation 1](#). Among them, C_h is larger than C_g and C_t , so it can be ignored.

$$C_{equal} = C_{touch} + C_{base} = C_t \parallel C_h \parallel C_g + C_p \approx C_t \parallel C_g + C_p \quad (1)$$

$$Sensitivity = \frac{C_{equal} - C_{base}}{C_{base}} = \frac{C_t \parallel C_h \parallel C_g}{C_p} \approx \frac{C_t \parallel C_g}{C_p} \quad (2)$$

The parallel plate capacitance is calculated by:

$$C = \epsilon_r \epsilon_0 \frac{A}{d} \quad (3)$$

Where:

- A = The contact area between the finger and the sensor pad covering layer.
- d = The thickness of the overlay.
- ϵ_0 = Air dielectric constant.
- ϵ_r = The dielectric constant of the overlay.

It can be seen from Equation 2 and Equation 3 that the methods to change the sensitivity are:

1. Reduce the thickness of the cover plate to increase the sensitivity.
2. Reduce the density of the grid ground, or increase the PCB thickness to reduce the sensitivity.
3. Connect the power ground to the earth ground to increase the sensitivity.
4. Increase the contact area A between the finger and the sensor pad covering layer to increase the sensitivity.

The electrode size cannot increase indefinitely. The main reason is that the maximum effective area of the parallel plate capacitance C_t is the same as the finger touch area. Therefore, C_t would also increase, resulting in a decrease in sensitivity.

2.2 Mutual-Capacitive Detection

As shown in Figure 2-4, mutual-capacitive capacitors use copper-clad double electrodes (receiving electrode Rx, sending electrode Tx) to detect the change in capacitance between the two electrodes. An advantage of mutual-capacitive detection is that the influence of C_p between sensor and power ground can be ignored.

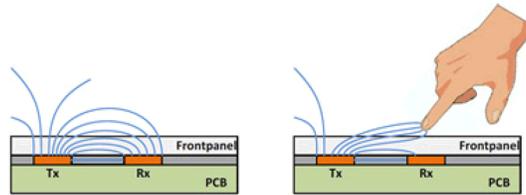


Figure 2-4. Mutual-Capacitive Detection

Take the simplest single button as an example, Figure 2-5 shows the detection model of mutual-capacitive capacitor. When touched by a human hand, the C_{RT} becomes two $2C_{RT}$, and C_{RTt} , C_t , C_h , and C_g are introduced at the same time. The capacitance between the two electrodes is **reduced**.

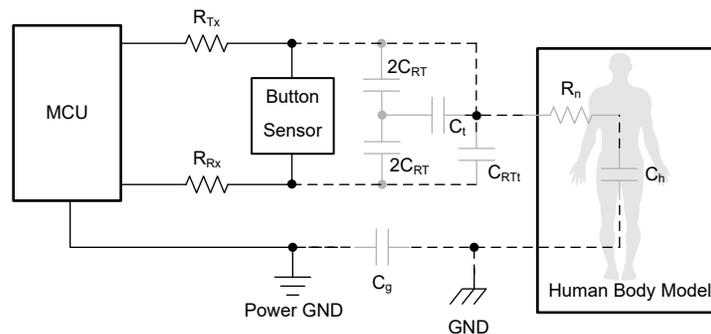


Figure 2-5. Mutual-Capacitive Detection Model

In Figure 2-5, the solid line indicates the PCB routing, and the dashed line indicates the touch-related routing. Gray components indicate equivalent capacitance or resistance.

C_{RTt} = The parallel capacitance between the Rx and Tx electrodes, which is introduced by a finger touch.

C_{RT} = The capacitance between the Rx and Tx electrodes, which is equivalently divided into two capacitances of $2C_{RT}$, when touched by a human hand.

The equivalent capacitance between Tx and Rx is shown in Equation 4. Sensitivity can be characterized as the ratio between the capacitance change caused by the touch and the base capacitance, as shown in Equation 5.

$$C_{equal} = \frac{4C_{RT}^2}{4C_{RT} + C_t \parallel C_h \parallel C_g} + C_{RTt} \approx \frac{4C_{RT}^2}{4C_{RT} + C_t \parallel C_g} + C_{RTt} \quad (4)$$

$$Sensitivity = \frac{C_{base} - C_{equal}}{C_{base}} \approx \frac{C_{RT} \times C_t \parallel C_g}{4C_{RT} + C_t \parallel C_g} - C_{RTt} \quad (5)$$

For mutual-capacitive touch, the main ways to increase the sensitivity are:

1. Reduce the thickness of the cladding layer
2. Increase the spacing between Tx and Rx

However, the sensitivity decreases if the finger cannot cover Tx and Rx at the same time.

Generally speaking, for self-inductance and mutual-inductance capacitance detection, the capacitance change caused by a finger touch is approximately 1 pF. But the base capacitance of self-inductance (capacitance value before touch) normally is higher than mutual-capacitance. Therefore, the sensitivity of mutual-capacitive is relatively higher, but it is also more susceptible to noise. From an application point of view, the self-inductance solution is more widely used due to its simple structure, and the mutual-capacitive solution is more used for matrix buttons. [Table 2-1](#) compares the two solutions.

Table 2-1. Comparison of Self-Capacitive and Mutual-Capacitive Detection

Features	Self-Capacitive Detection	Mutual-Capacitive Detection
Difficulty in design and wiring	Simple	Complicated
Affected by the grounded metal shell	Yes	No
Affected by the ungrounded metal shell	No	Yes
Passed CNI test (EMC) level based on the button structure	High level	Low level
Support high-density buttons	No	Yes
Waterproof and anti-fog performance	Low	High
Support metal touch	Yes	No
Wheel or slider performance	High resolution	Low resolution
Realized proximity sensing distance	>10 cm	<3-4 cm
Support touch screen function	No	Yes

2.3 TI's Capacitive Touch Technology

TI's CapTIvate capacitive touch sensing technology is based on charge transfer collection. The operation includes 1) charging the sensor input capacitor C_{equal} and 2) transferring the accumulated charge to the internal sampling capacitor C_{sample} .

This process repeats until the voltage on both sides of C_{sample} reaches the trigger voltage V_{trip} of the internal comparator. The number of charge transfers required to reach the threshold directly characterizes the size of C_{equal} . When the capacitive sensor is touched by a human hand, C_{equal} and charge transfer values change. The MCU senses the occurrence of a touch event by comparing the numbers of different charge transfer cycles. The MSP430 MCU uses a current mirror to control the proportional relationship between the input current of C_{sample} and the discharge current of C_{equal} , to equivalently amplify C_{sample} and have a larger range.

For self-capacitive detection, C_{equal} is equal to the capacitance between ground and RX I/O port. Through the charge and discharge phase shown in [Figure 2-6](#), the electric charge in C_{equal} is transferred to the internal C_{sample} . For mutual-capacitive detection, the circuit structure is more complex than that of the self-capacitive detection, but the principle is the same. By keeping the voltage of capacitance between the ground and RX I/O port to be the same in the charge and discharge phase, only the electrical charge is transferred to the internal C_{sample} .

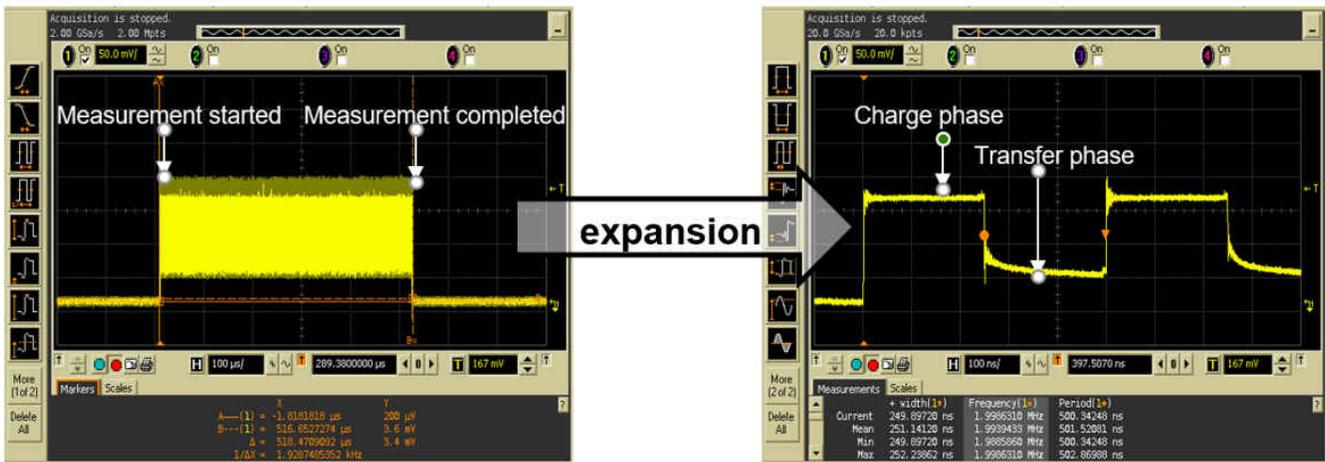
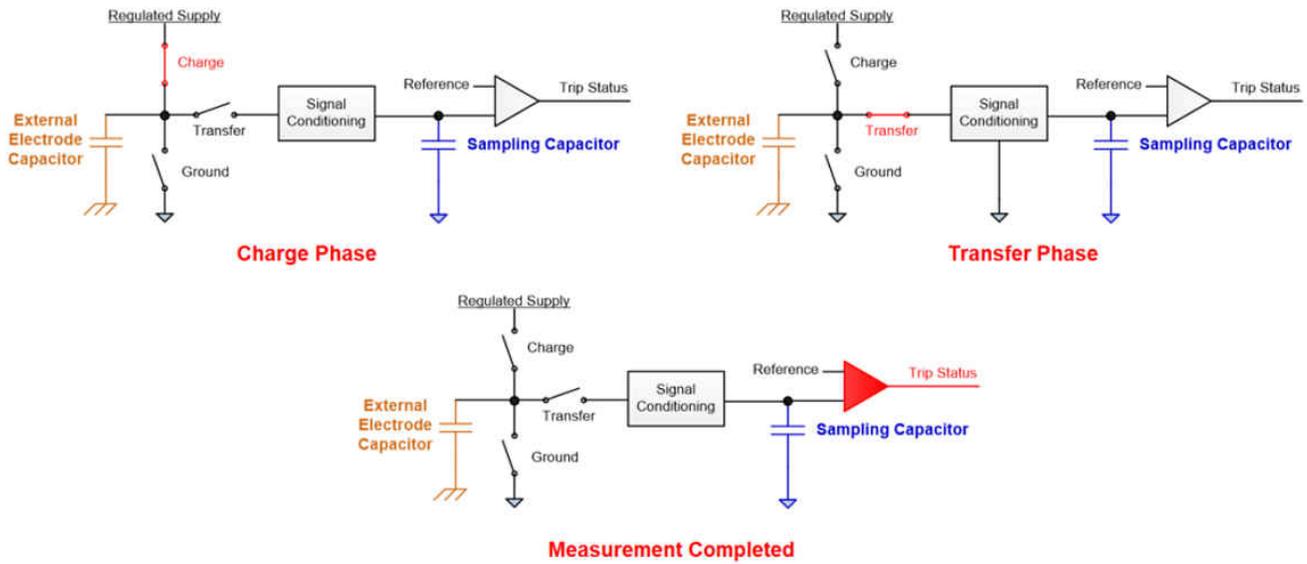


Figure 2-6. Principle of Self-Capacitive Detection

3 MCU Selection and Evaluation

3.1 Determine Sensor Requirements

Before starting a capacitive touch project using an MSP430 MCU, you must understand the selected sensor type, the number of sensors, and the occupied CAP I/O port. Finally, select the appropriate MSP430 MCU according to these requirements.

Capacitive touch sensors are mainly divided into the following types: button or proximity sensor, slider or wheel, and touch screen.

3.1.1 Button or Proximity Sensor

Both buttons and proximity sensors are zero-dimensional sensors. The principle of both implementations is the same. The difference is that proximity sensing tends to have higher sensitivity and a larger sensor enclosing area.

Self-capacitive button:

As [Figure 3-1](#) shows, the self-inductive button is mainly composed of copper clad connected to the CAP I/O pins and an insulating covering material. The button is also surrounded by a grounded copper clad separated from the button pad by a ring gap. Each button requires 1 MCU CAP I/O pin.

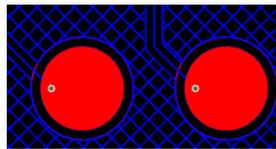


Figure 3-1. Self-Capacitive Button PCB

Mutual-capacitive button:

In applications that require a large number of buttons, capacitive sensors can be arranged in a matrix, and each button is connected to 2 CAP I/O pins, with 1 CAP I/O pin connected to each row or column of buttons. As shown in [Figure 3-2](#), the two box-shaped copper clad inside and outside are Tx and Rx. The program supports multi-button touch. X rows by Y columns require X+Y MCU CAP I/O pins.

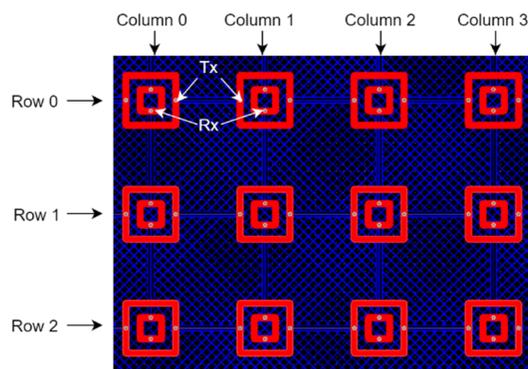


Figure 3-2. Mutual-Capacitive Button Matrix PCB

Self-capacitive proximity sensor:

Typically, proximity sensors use the self-capacitive scheme. The main reasons are that self-capacitive detection has a longer detection distance than mutual-capacitive detection, more flexible wiring and structure, and a higher signal-to-noise ratio. The PCB outer circle in [Figure 3-3](#) shows the proximity sensor. In addition to using PCB copper to form a proximity sensor, wires or even conductive metal structures can also be used. A self-inductive proximity sensor requires 1 MCU CAP I/O pin.

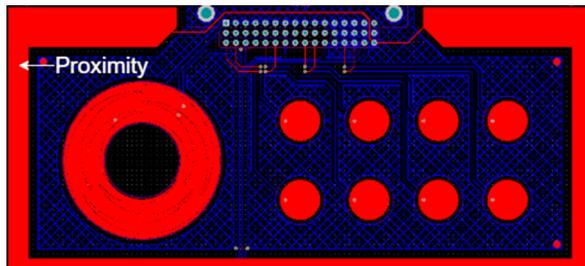


Figure 3-3. Mutual-Induction Proximity-Sensing PCB

3.1.2 Slider or Wheel

Both the slider and the roller are one-dimensional linear sensors, and their implementation principles and structures are similar. Because the base capacitance of mutual-capacitive detection is low and the signal fluctuates more than in self-capacitive detection, mutual-capacitive detection is less often used in sliders or wheels. Therefore, only the self-inductance slider or wheel is discussed, as shown in Figure 3-4.

The basic structure of the slider or wheel is the buttons. By interlacing the buttons, when the human hand moves, because the contact area of the two adjacent buttons is different, the movement of the signal can be detected. In addition, the two copper coatings at the beginning and the end of the slider are connected together by wires and belong to the same button.

For self-capacitive sliders and scroll wheels, at least 3 MCU CAP I/O pins are required. The resolution range of the sliders or scroll wheels supported by the software is 3 to 65535. The actual number of CAP I/O pins occupied and the selected resolution must be adjusted according to the requirements of the application.

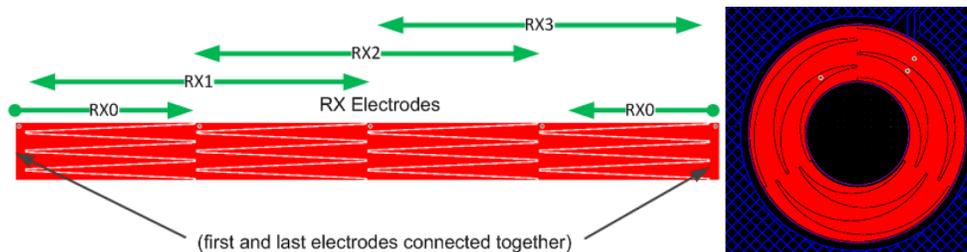


Figure 3-4. Self-Capacitive Slider or Wheel PCB

3.1.3 Touch Panel

The CapTIvate touch panel is a two-dimensional sensor based on mutual capacitance. The detection points are formed by the intersection of rows and columns of a diamond pattern. Its basic structure is similar to matrix buttons. By interlacing the buttons, a gradual change in the signal can be detected on two adjacent buttons when the human hand moves. In Figure 3-5, the intersection of 4 RX and 4 TX forms 16 measurement nodes. A single touch panel often requires more than 8 MCU CAP I/O pins.

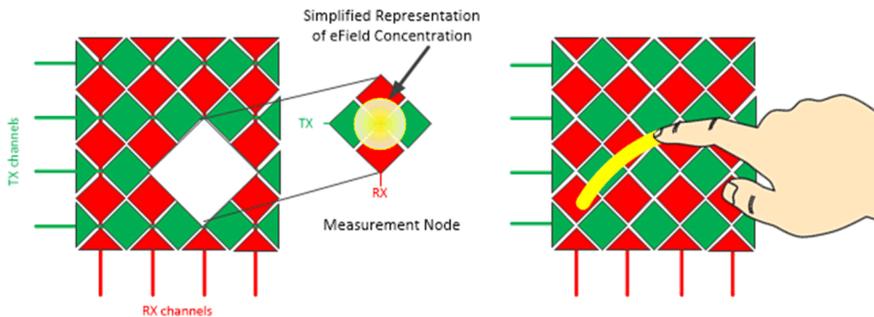


Figure 3-5. Touch Panel

3.2 MCU Selection

Table 3-1 summarizes all of the MSP430 MCUs that support CapTIvate capacitive touch. According to the required CAP I/O pins and other peripherals, the appropriate MCU can be selected. The capacitive touch code occupies 3KB to 6KB depending on the configuration, and the maximum output current of the I/O port with a 3-V power supply is 5 mA.

Table 3-1. MSP430 CapTIvate MCU Selection

Part Number	Generation	CAP I/Os	FRAM (KB)	RAM (KB)	ADC	Communication	Package
MSP430FR2512	Gen1	4	8	2	10 bit	1 UART, 1 I2C, 2 SPI	TSSOP 16, VQFN 20
MSP430FR2522	Gen1	8	8	2	10 bit	1 UART, 1 I2C, 2 SPI	TSSOP 16, VQFN 20
MSP430FR2532	Gen1	8	8	1	10 bit	2 UART, 1 I2C, 3 SPI	VQFN 24
MSP430FR2533	Gen1	16	16	2	10 bit	2 UART, 1 I2C, 3 SPI	TSSOP 32, VQFN 32
MSP430FR2632	Gen1	8	8	2	10 bit	2 UART, 1 I2C, 3 SPI	DSBGA 24, VQFN 24
MSP430FR2633	Gen1	16	16	4	10 bit	2 UART, 1 I2C, 3 SPI	DSBGA 24, TSSOP 32, VQFN 32
MSP430FR2672	Gen2	16	8	2	12 bit	2 UART, 2 I2C, 4 SPI	VQFN 32
MSP430FR2673	Gen2	16	16	4	12 bit	2 UART, 2 I2C, 4 SPI	VQFN 32
MSP430FR2675	Gen2	16	32	6	12 bit	2 UART, 2 I2C, 4 SPI	LQFP 48, VQFN 32, VQFN 40
MSP430FR2676	Gen2	16	64	8	12 bit	2 UART, 2 I2C, 4 SPI	LQFP 48, VQFN 32, VQFN 40

The CapTIvate MCUs include two generations of devices. The first-generation product (Gen1) focuses on cost performance, while the second-generation product (Gen2) has more abundant peripherals and stronger CapTIvate module performance. Table 3-2 compares the two device generations.

Table 3-2. Comparison of Gen1 and Gen2

Features	Gen1 Device	Gen2 Device	Gen2 Advantages
Electrode charging voltage	VREG power supply: 1.5 V	VREG power supply: 1.5 V DVCC power supply: 2.7 V to 3.6 V	Will affect the charging and discharging voltage, the newly added DVCC mode can improve the signal-to-noise ratio and anti-interference ability.
Input bias current	No	Yes	Improve anti-interference ability.
Conversion and anti-noise processing	Software	Hardware	Improve response speed and save code space.
Typical signal-to-noise ratio (-40°C to 25°C)	19:1 to 36:1	28:1 to 42:1	Higher signal-to-noise ratio and lower temperature drift.

3.3 EVM Development Board Selection and Evaluation

The entire development chain is made up of a GUI, an IDE, an SDK, a programmer, an MCU, and a sensor board. Part of the MCU board and sensor board are integrated. Table 3-3 lists the combinations of hardware and software that can be used.

Table 3-3. CapTIvate Development Chain

GUI	IDE	SDK	Programmer	MCU Board	Sensor Board	Supported Sensor Type
CapTIvate Design Center	CCS, IAR	MSPWare	CAPTIVATE-PGMR	CAPTIVATE-FR2676, CAPTIVATE-FR2633	CAPTIVATE-BSWP	Self-mode button, wheel, slider, proximity
					CAPTIVATE-PHONE	Mutual-mode button, wheel, slider, proximity
				BOOSTXL-CAPBUTTONPAD (MSP430FR2522)		12 mutual-mode buttons and 1 proximity
				EVM430-CAPMINI (MSP430FR2512)		4 self-mode buttons

The software parts of the development chain are:

- GUI: Used to generate code and debug parameters online.
- IDE: Used to debug code.
- SDK (optional): Software development kit, used to provide driverlib to develop custom functions, which is not mandatory to install.

The hardware parts of the development chain are:

- Programmer: Supports JTAG and USB HID to UART or I2C communication. Used for programming code and capacitive touch online parameter adjustment.
- MCU board: MSP430 single-chip evaluation board.
 - CapTivate-FR2676 is used to evaluate Gen2 solutions, or the MSP430FR267x series.
 - CapTivate-FR2633 is used to evaluate Gen1 solutions, or the MSP430FR263x and MSP430FR265x series.
 - BOOSTXL-CAPBUTTONPAD is only used to evaluate specific 12-button panel solutions or to evaluate the MSP430FR2522.
 - EVM430-CAPMINI is only used to evaluate specific 4 self-inductive button schemes or to evaluate the MSP430FR2512.
- Sensor board: An evaluation board containing various sensors.
 - CAPTIVATE-BSWP is used to evaluate self-inductive solutions.
 - CAPTIVATE-PHONE is used to evaluate mutual-capacitive solutions.

For additional materials related to CapTivate technology, including instruction documents, teaching videos, development boards, and application manuals, refer to the *Capacitive Touch Sensing* section of the [MSP430™ MCUs Development Guide Book](#).

TI offers GUI and CCS projects for all EVM boards. The default installation directory is:

C:\Users\UserName\CapTivateDesignCenter_x_xx_xx_xx \CapTivateDesignCenterWorkspace\TI_Examples

Refer to the steps in [Section 6](#) to complete the evaluation of the EVM, which also includes the specific software installation and the use and debugging of the EVM.

4 Mechanical Structure and Hardware Design

As the first step of capacitive touch design, the mechanical structure and hardware design greatly affect the sensitivity and noise immunity of the capacitive touch solution. Therefore, before reading this chapter, first read [Section 2](#) to understand the basic principles of capacitive touch. In general, the mechanical structure and hardware design are mainly trade-offs between structural constraints, sensitivity, and noise immunity. This means that it is often necessary to repeatedly modify a design based on test feedback.

4.1 Mechanical Structure Design

4.1.1 Cover Layer Design

In the design of capacitive touch, capacitive sensors are placed under the cover layer to reduce environmental impact and prevent ESD problems caused by direct finger contact. It can be seen from [Equation 6](#) that the dielectric constant and thickness of the covering material affect the sensitivity of the capacitive touch. [Table 4-1](#) lists the dielectric constants of different materials.

Table 4-1. Dielectric Constant of Covering Material

Material	ϵ_r
Air	1.00059
Glass	4 - 10
Sapphire glass	9 - 11
Mica	4 - 8
Nylon	3
Plexiglas	3.4
Polyethylene	2.2
Polystyrene	2.56
Polyethylene terephthalate (PET)	3.7
FR4 (fiberglass and epoxy)	4.2
PMMA (polymethyl methacrylate)	2.6 - 4

For panels made from different materials, the touch capacitance can be calculated according to the following equation:

$$C = \epsilon_r \epsilon_0 \frac{A}{d} = A \epsilon_0 \sum \frac{\epsilon_{ri}}{d_i} \quad (6)$$

Where:

- A = The contact area between the finger and the sensor pad covering layer.
- d_i = The thickness of the different covering layers.
- ϵ_0 = The dielectric constant of air.
- ϵ_{ri} = Dielectric constants of different covering layers.

Because the dielectric constant of air is approximately 1, it can be seen from [Equation 6](#) that the presence of the variable phase thickens the panel, so air between the sensor and the cover layer should be eliminated as much as possible. In many cases, the sensitivity of the buttons is low because the panel is too thick. Because the conductor interferes with the electric field mode of capacitor charging and discharging, conductive materials and paint or adhesives containing metal particles cannot be used. Typical recommended mixtures are 3M™ 200MP, 467MP, and 468MP.

4.1.2 Sensor Structure Selection

Various materials can be used to construct capacitive sensors. The following sections briefly introduce common sensor structures.

4.1.2.1 Copper-Clad Sensor (PCB)

This solution uses copper pads etched on the surface of the PCB as sensors, which is the most common implementation method for capacitive touch. Rigid PCBs are the most widely used and are suitable for flat front panels, while flexible PCBs are suitable for curved front panels. Flexible PCB is thinner than rigid PCB, which means that the distance between the button and the ground plane is relatively short. Therefore, the density of the grid ground plane should be appropriately reduced.

4.1.2.2 Conductive Washer or Spring Sensor

Conductive washer or spring sensors allow the expansion of the touch button space through washers or springs made of conductive material. These sensors are suitable when the touch front panel cannot be attached to the PCB, or for curved, inclined, or irregular front panels.

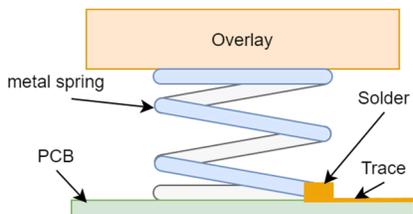


Figure 4-1. Diagram of Spring Structure

4.1.2.3 Electronic Ink Sensor

Electronic ink sensors use conductive ink to form the capacitive sensor. Close integration with the front panel is achieved by spraying or printing. The main features are flexible patterns and shorter design cycles, and these sensors are suitable for irregular front panels. Another application scenario is the ITO liquid crystal display, which uses the transparency characteristics of the ITO film. Because the film resistance of electronic printing ink is larger than that of copper, the series resistance must be reduced.

4.1.3 Mechanical Design Checklist

The main aspects of the mechanical structure that affect the touch sensitivity are coating, housing, and surrounding devices. Coating design mainly affects the capacitance change caused by finger touch. For a self-inductive solution, grounding the shell improves the noise immunity of the product, but it increases the capacitance to ground and reduces the sensitivity. For a mutual-capacitive solution, the grounding of the surrounding environment has little effect on the sensitivity. As capacitive touch is a sensitive device and also a source of interference, a grid ground plane is required for protection and isolation. [Table 4-2](#) is a checklist of mechanical structure design.

Table 4-2. Mechanical Structure Design Checklist

Number	Component	Recommendation
1	Overlay	Material
2		Thickness
3		Cascade
4	Metal shell	In the case of ensuring sensitivity, the shell should be grounded as much as possible to improve noise immunity.
5	Surrounding devices	Keep the PCB with button function as far away as possible from any internal radiation noise source in the product. Keep the PCB with button function as far away as possible from sensitive devices. Keep the PCB with button function as far away as possible from the PCB with a lot of ground.

4.2 Hardware Design

4.2.1 Schematic Design

The main concern of the schematic design is both to ensure that the MCU function is normal and to maximize EMC anti-noise performance. [Table 4-3](#) is a checklist for schematic design. For specific design solutions, refer to the device-specific data sheet, or refer to the EVM schematic diagram in [Section 3.3](#).

Table 4-3. Schematic Design Checklist

Number	Classification	Component	Recommendation	
1	Smallest system	Reset and programming circuit	Add 47-k Ω pullup resistor and 1-nF pulldown capacitor to Reset pin.	
2		Power supply circuit	Add 10- μ F and 0.1- μ F capacitors to VCC and GND, and place them close to the MCU.	
3	Capacitive touch	VREG filter circuit	Add 1- μ F capacitance to ground close to VREG pin, ESR \leq 200 m Ω .	
4		Series resistance on CAP I/O	Add a 470- Ω to 10-k Ω resistor close to the MCU pin for ESD protection and anti-noise filtering.	
5		CAP I/O pin assignment (if possible)	It is recommended to use the "Auto-Assign" function of CapTIvate Design Center to assign CAP I/O pins. In addition to buttons and proximity sensors, scroll wheels, sliders, and touch panels have sequence requirements for scanning the CAP I/O pins. The Auto-Assign function includes this sequence to assign scroll wheels, sliders, and touch panel CAP I/O pins after configuring the buttons.	
6	EMC anti-noise (Optional)	EMC filter capacitor	For mutual-capacitive applications, add 68 pF of capacitance to ground between the series resistance on the RX pin and the sensor electrode.	
7		TVS diode	Add a 3.3-V TVS tube with low leakage and low parasitic capacitance between the CAP I/O series resistance and the electrode. Add a general TVS tube to the power supply and external connection line.	
8		Common mode inductors/magnetic beads	Add common mode inductance and magnetic beads to the power supply as needed.	
9	Others	I2C communication line pullup resistor	Add 2.2-k Ω pullup resistor.	
10		I2C communication pins (GUI default configuration)	MSP430FR25x2: P2.4: IRQ (OPEN DRAIN) P2.5: UCB0 I2C SDA P2.6: UCB0 I2C SCL	MSP430FR263x, MSP430FR253x, MSP430FR267x: P1.1: IRQ (OPEN DRAIN) P1.2: UCB0 I2C SDA P1.3: UCB0 I2C SCL
11		UART communication pin (GUI default configuration)	MSP430FR25x2: P2.0: UCA0 UART TXD P2.1: UCA0 UART RXD	MSP430FR263x, MSP430FR253x, MSP430FR267x: P1.4: UCA0 UART TXD P1.5: UCA0 UART RXD
12		Boot Loader (BSL)	For BSL pin definition, refer to the Bootloader chapter in the corresponding device datasheet.	
13		Button to MCU connector	Will increase the parasitic capacitance to the ground, it is not recommended to use.	
14	Test point	Increase the related test points of VCC, GND, communication port.		

During preliminary evaluation of the EMC anti-noise design, TI recommends allowing for changes to components and layout and finalizing according to the test results. An important choice is the series resistance on the CAP I/O pins. Because the series resistance affects the base capacitance and the capacitance change caused by the

touch at the same time, the series resistance itself does not affect the sensitivity of the button. Because it forms a low-pass filter with parasitic capacitance, a larger series resistance can increase the noise immunity of the button. However, too large a resistance affects the charge transfer time. The charge and discharge waveforms on the CAP I/O pins are shown in [Figure 4-2](#). This change to the charge transfer time decreases the sensitivity of the button, which is especially obvious in a self-inductive solution with a larger base capacitance of the sensor. The charge transfer period can be extended and the Frequency Divider parameter in the GUI can be modified to solve the problem.

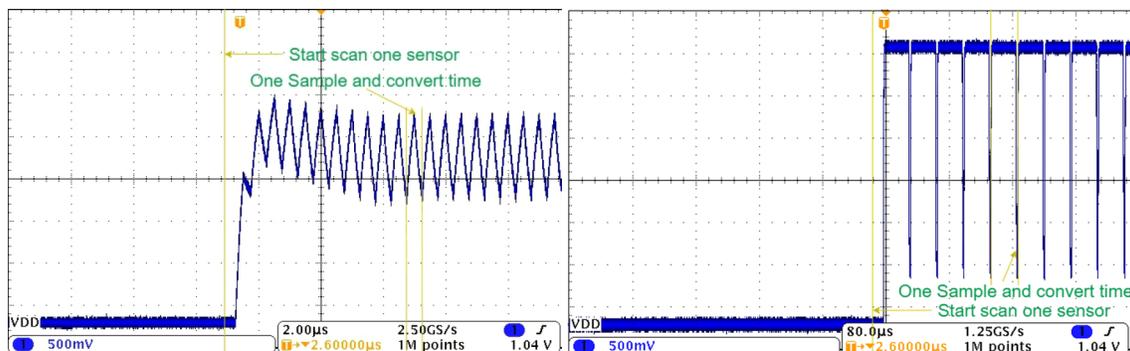


Figure 4-2. Incomplete and Complete Charge Transfer Cycle (GEN2)

4.2.2 PCB Layout

Some of the goals of PCB layout and mechanical structure design are the same, to improve signal strength and to reduce EMC problems. [Table 4-4](#) is the PCB layout checklist. For specific examples, refer to TI EVM board [CAPTIVATE-BSWP](#) (self-inductive) and [CAPTIVATE-PHONE](#) (mutual-capacitive).

Table 4-4. PCB Layout Checklist

Number	Component	Recommendation	
1	Button (self-capacitance)	Shape	Solid circle or square.
		size	The diameter/side length is 10 mm to 12 mm.
		Distance to ground grid copper	The button is placed on the top layer of the PCB, and the ground copper is 0.5 times the thickness of the cover layer of the button.
2	Button (mutual capacitance)	Shape	Square or round, inner Rx, outer Tx.
		Size	Outer diameter/side length is 10mm and 12mm. RX width is 0.5 mm, Tx width is 1 mm, and the distance between Rx and Tx is 0.5 mm.
		Distance to grid copper	The button is placed on the top layer of the PCB, and the ground copper is 0.5 times the thickness of the cover layer of the button.
3	Slider or wheel (self-capacitance)	Shape	Refer to Automating Capacitive Touch Sensor PCB Design Using OpenSCAD Scripts
		Size	The electrode width is 10 to 12 mm, and the length depends on the required touch area.
		Number of electrodes	More than or equal to 3 electrodes, add electrodes according to the sensitivity and resolution requirements.
4	Proximity sensor (self-capacitance)	Shape size	Solid copper clad, hollow copper clad or wire, the sensing distance is positively related to the enclosed area.

Table 4-4. PCB Layout Checklist (continued)

Number	Component		Recommendation
5	Sensor wiring	Width	8 mil or the minimum thickness allowed by the PCB manufacturer.
		Length	Minimize the length from the sensor to the controller.
Points to note when routing		Keep 10 mil to 20 mil spacing with grounding copper. No sharp angles. Reduce via holes, via hole diameter 10 mil. Keep away from other touch signal wires, digital signal wires and analog signal wires by more than 4mm or choose vertical routing.	
7	Ground copper		Use grid-like copper plating (line width 8 mil, grid width 64 mil, angle 45°). Self-capacitive design, double-sided copper-clad, selective copper-clad directly under the button. Mutual-capacitive type can use copper plating on only the bottom surface of the board.
8	Series resistance, TVS tube, EMC filter capacitor placement		Place within 10 mm of MCU pin.
8	Moisture and liquid resistance		Self-inductive solution: Reduce the grounding copper and improve the touch sensitivity to reduce the parasitic capacitance change caused by water. Mutual-capacitive scheme: Liquid Tolerant Capacitive Touch Buttonpad Reference Design

In terms of sensitivity, buttons and slider wheels that are similar in width to a finger, and sensor traces that are narrow and short can enhance the capacitance change caused by touch. Fewer vias and thinner copper can reduce the capacitance to ground. However, grounding copper also directly affects EMC anti-noise ability. The specific layout and grid density must be adjusted according to test results. Because capacitive touch is both a source of interference and a sensitive device, for EMC considerations, in addition to adding protection devices, route touch signals away from other signal lines or devices.

5 Software Design and Parameter Tuning

Software development for a capacitive touch application is complicated and requires software tools such as GUI and IDE. The development process includes setting parameters and function development. Therefore, before reading this chapter, read [Section 3.3](#) and [Section 6](#) to understand the entire capacitive touch development chain, and to be familiar with the basic operations of the GUI and IDE. For the information on the MSP430 MCU itself, refer to the [MSP430™ MCUs Development Guide Book](#).

Capacitive touch software development can be roughly divided into five parts as shown in [Figure 5-1](#). From the perspective of development time, software tuning takes the longest time. The following sections describe these development phases.

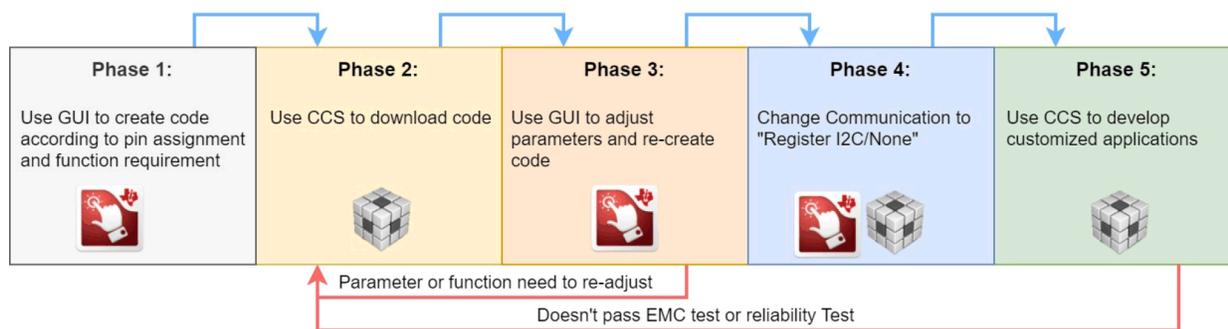


Figure 5-1. Software Design Process

5.1 Concepts Required for CapTlvate Software Development

Before entering the various stages of development, the following section describes important concepts of the CapTlvate module to help users understand CapTlvates principles and the correlation between parameters. For more information, refer to the [Technology](#) chapter in the [CapTlvate Technology Guide](#).

5.1.1 CapTlvate Module and GUI Function Description

The CapTlvate module consists of a Block and a CapTlvate Core. The Block is responsible for scanning the external capacitance. The CapTlvate Core is responsible for the scanning control of the block and the anti-noise function. One Block corresponds to four CAP I/O pins. This means that only one CAP I/O at a time can be scanned in each block, so optimal configuration of the button scanning sequence can save a certain amount of time.

In the CapTlvate GUI, the MCU widget and sensor widget are used to configure the CapTlvate module. The relationship between the GUI and the CapTlvate module is shown in [Figure 5-2](#). The sensor widget is mainly responsible for the control of the Block. The MCU widget is mainly responsible for the control of the CapTlvate Core.

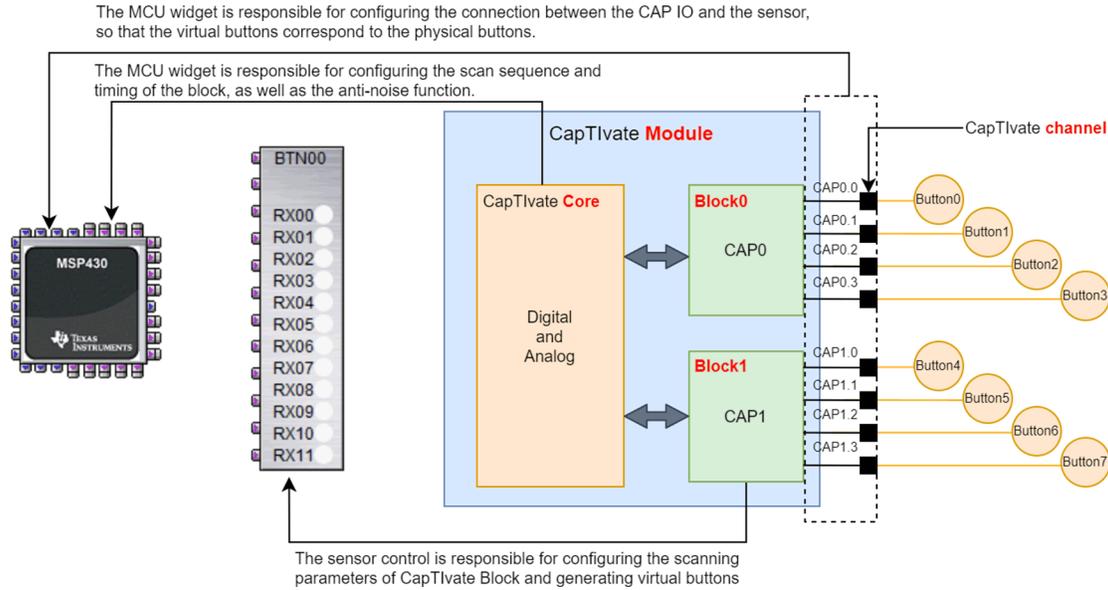


Figure 5-2. Correspondence Between the GUI and the CapTlvate Module

5.1.2 MCU Working Mode

The CapTlvate MCU has two working modes: Active mode and Wake-on-Prox mode. For the power consumption in different modes, refer to the device-specific data sheet. The mode is configured in the MCU widget. As shown in Figure 5-3, in Active mode, the CPU controls the CapTlvate peripheral to perform sensor scanning, and then the CPU enters the low-power mode after sleeping. After waiting for "Active Mode Scan Rate" (ms), the Timer in the CapTlvate peripheral wakes the CPU to scan the sensor again, and this process repeats.

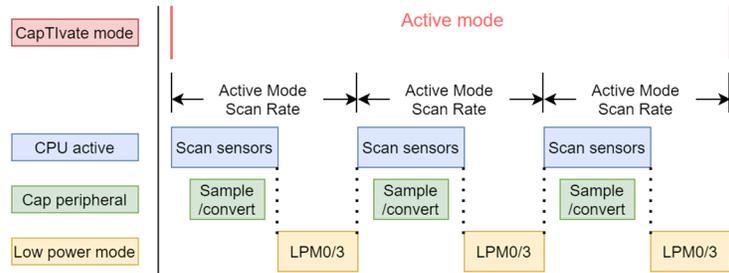


Figure 5-3. MCU Status in Active Mode

The Wake-on-Prox mode uses the Cap peripheral's ability to work without the CPU, and achieves lower power consumption than the Active mode by turning off the CPU for a long time. Because no CPU is involved, the Cap peripheral register configuration cannot be changed, so there can only be one proximity wake-up sensor scanned.

The workflow is shown in Figure 5-4. When the MCU is powered on, it works in Active mode, and multiple buttons are scanned in turn. If no touch occurs, the MCU waits for the "Inactivity Timeout" sensor scan period, then changes to Wake-on-Prox mode. The CPU turns off, and the peripheral scans the wake-up sensor at the "Wake-on-Prox Mode Scan Rate" interval. The MCU can enter LPM4 to further reduce power consumption.

After waiting for the "Wakeup Interval" scan interval (usually a few minutes), the MCU wakes up, enters Active mode, and updates the parameters of all buttons to adjust for environmental drift. After the "Inactivity Timeout" count, the MCU enters Wake-on-Prox mode again. This cycle repeats.

In Wake-on-Prox mode, if the proximity wake-up sensor detects a proximity event or if the detection signal exceeds the Error Threshold, the MCU wakes up and enters Active mode.

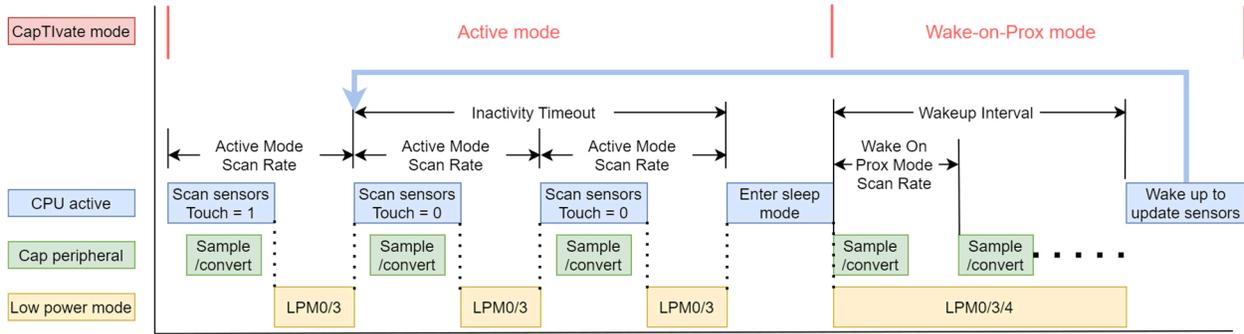


Figure 5-4. MCU Status in Wake-on-Prox Mode

5.1.3 Relationships of Parameters in CapTivate Technology

The parameter structure of CapTivate technology is shown in Figure 5-5. The parameters are divided into hardware configuration parameters and software configuration parameters, all of which are configured in the sensor widget. From this, you can also know which parameters can be automatically updated when the CPU is not working and the CapTivate MCU is in Wake-on-Prox mode.

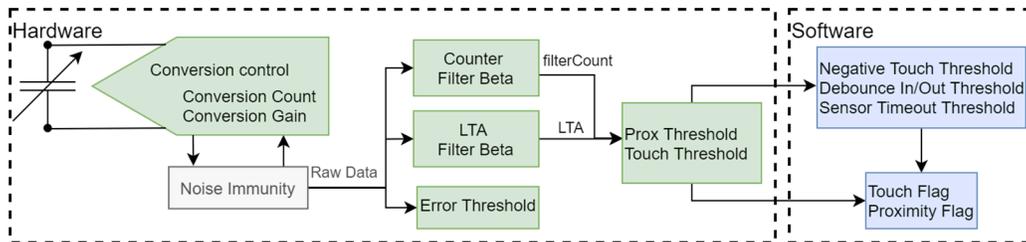


Figure 5-5. Parameter Relationships

First, the sampling module generates raw data by detecting the external capacitance. Conversion Count and Conversion Gain determine the gain of the sampling module. Then the data passes through the optional anti-noise module (sampling frequency spread spectrum, oversampling, and other functions) to achieve noise filtering. For GEN1, part of the anti-noise function is implemented by software.

The output data is first compared to the Error Threshold. Then filterCount (default is 1) is generated through IIR filtering of different intensities, which is used to characterize real-time capacitance changes, and LTA (Long time average, default is 7) is used to characterize the base capacitance of the environment. Here filterCount corresponds to "Count" in the GUI data monitoring module, and LTA corresponds to "LTA". The Delta between the two is used to characterize the change of capacitance generated by a touch. The relationship between the capacitance change percentage caused by the touch and the filterCount and LTA is shown in Equation 8. For self-capacitance detection, Delta is a positive value, and for mutual-capacitance detection, Delta is a negative value.

$$\Delta = filterCount - LTA \quad (7)$$

$$\Delta C_{touch} = Gain \left(\frac{1}{LTA + \Delta} - \frac{1}{LTA} \right) * 100 \% \quad (8)$$

The entire signal change process is: when power is on, the calibration function calibrates the equivalent value of the environmental base capacitance to Conversion Count. That is, LTA = Conversion Count. Therefore, Conversion Count can also be understood as a parameter that determines the resolution of the system. At this time, because there is no touch, filterCount=LTA. When touched by a human hand, due to the different filter strengths of the LTA Filter and the Counter Filter, the filterCount changes rapidly, and the Delta increases from 0. When the Prox Threshold is triggered, the LTA Filter is closed and the LTA value remains unchanged. filterCount continues to change, Delta continues to increase, which triggers Touch Threshold. When the hand leaves, if the signal is weaker than the Prox Threshold, the LTA Filter opens and the LTA value starts to refresh. Therefore, the Prox Threshold must be smaller than the Touch Threshold to ensure the normal working mechanism of LTA. In

addition, if the Prox Threshold is set too large, human touch causes the LTA to change significantly, and the LTA drifts after multiple consecutive touches, and the system cannot correctly respond to subsequent touches.

Strictly speaking, both Prox Threshold and Touch Threshold are thresholds set for Delta. When triggered, the corresponding global variable of the system is set, and the signal is sent to the subsequent program logic. It should be noted that Prox Threshold is an absolute threshold, while Touch Threshold is a relative threshold. The direct relationship between the two and LTA is shown in Equation 9 and Equation 10. In addition, it can be seen from Equation 8 that Delta has a parallel relationship with ΔC_{touch} . This is also the reason why Prox/Touch Threshold and Prox/Touch Threshold Percentage are different, because the latter directly characterizes the change in touch capacitance.

$$|\Delta| = \text{Prox Threshold} \tag{9}$$

$$|\Delta| = LTA \times \frac{\text{Touch Threshold}}{128} \tag{10}$$

Other Thresholds determines the setting events of the Touch flag and Proximity flag of the entire Sensor. For example, the MCU is powered on when a human hand touches. Then, the touch signal of the human hand is counted as the base capacitance of the environment. At this time, removing the finger generates a negative signal. When the negative signal reaches the Negative Threshold, system calibration is triggered. The debounce in and out threshold introduces a signal anti-jitter mechanism to combat noise, which delays the response of the button to a certain extent. The Sensor Timeout Threshold is used to prevent environmental changes from triggering the Prox Threshold situation. After the threshold is triggered, the system recalibrates the environmental base capacitance.

5.1.4 Object Structure in CapTlvate Technology

There are four types of objects in CapTlvate, Top Level, Sensor, Cycle, and Element. Figure 5-6 shows the parent-child relationship between different structures, and also gives the object name, structure name, and actual naming.

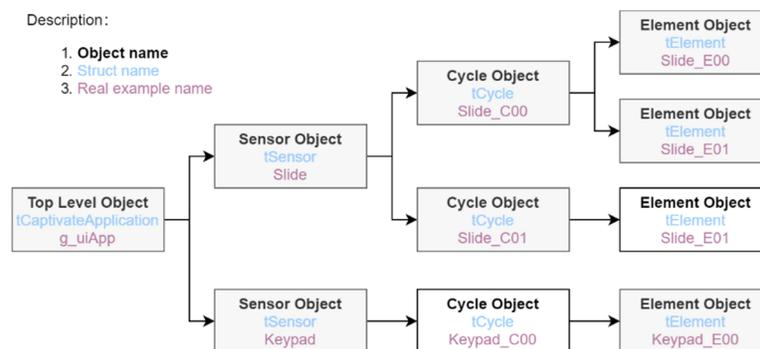


Figure 5-6. Object Structure in CapTlvate Technology

Top Level is used to describe the working mode of MCU. It belongs to the abstract concept of MCU. It corresponds to the MCU widget in the GUI, and the parameters configured by the MCU widget hang under the Top-Level structure. This part of the content is in 5.1.1 of this chapter. Discussed in the subsection, Sensor, Cycle, and Element correspond to the sensor widgets in the GUI. The parameters configured by the sensor widgets are placed under these three structures. This part of the content is also discussed in Section 5.1.2.

Sensor corresponds to the five sensor forms of button panel, proximity sensor, scroll wheel, slider, touch panel, and proximity sensor, which belong to the abstract concept of physical sensors, so there can be multiple Sensor objects in a system.

Because one capacitance detection module of CapTlvate MCU corresponds to 4 CAP IOs, it is necessary to select the scanned IO ports sequentially by chip selection, so Cycle corresponds to each round of scanning.

Element is an abstract concept of physical buttons. All Sensor modes, button panels, proximity sensors, scroll wheels, sliders, and touch panels are essentially based on the signals measured by each button, and then processed on the software to obtain the corresponding zero-dimensional, one-dimensional, and two-dimensional

information. Therefore, the button is the most basic structure in the sensor, and the Element is also the most basic object.

5.1.5 CapTIvate MCU Communication Modes

CapTIvate supports four communication modes based on UART and I2C: UART, BUCK_I2C, Register_I2C, and None. The default baud rate of UART is 250 kbps. The default address of I2C is 7-bit 0x0A, and the maximum support speed is 400 kHz.

- **None:** Disable the communication module. Customers need to develop communication protocols to communicate with Host MCU by calling library functions. To save additional code space, choose register-level code in MSPWare as a reference for development. When Noise Immunity is not enabled, the default project takes about 2.4k of code space. After noise immunity is enabled, 4.8k of code space is occupied.
- **UART, BUCK_I2C:** Communicates with CapTIvate Design Center through UART or I2C for adjustment of capacitive touch by the GUI. Supports richer communication protocols, and the bus transmits a large amount of low-level information, which is generally used for online parameter adjustment. The UART communication module occupies approximately 1.5KB of code, and the BUCK_I2C communication occupies approximately 2KB of code.
- **Register_I2C:** Communicates with the host MCU through I2C for adjustment of capacitive touch parameters by the host MCU. Fewer debugging commands are supported, and code occupies approximately 1.4KB of code.

The intended process is that the user first selects the UART or BUCK_I2C communication mode to use the GUI to complete the parameter debugging, and then changes the communication mode to None, and develops the protocol for the host computer communication by themselves.

5.2 Phase 1: GUI Configuration

Follow the instructions below to configure the relevant parameters. For more information, refer to [Design Center GUI](#) section in the user guide.

5.2.1 GUI Main Interface

The main interface of the GUI is shown in [Figure 5-7](#). The operation of this interface is mainly to create a project and generate the corresponding virtual sensor. The parameters to be configured are shown in [Table 5-1](#). Other parameters are involved in Phase 2.

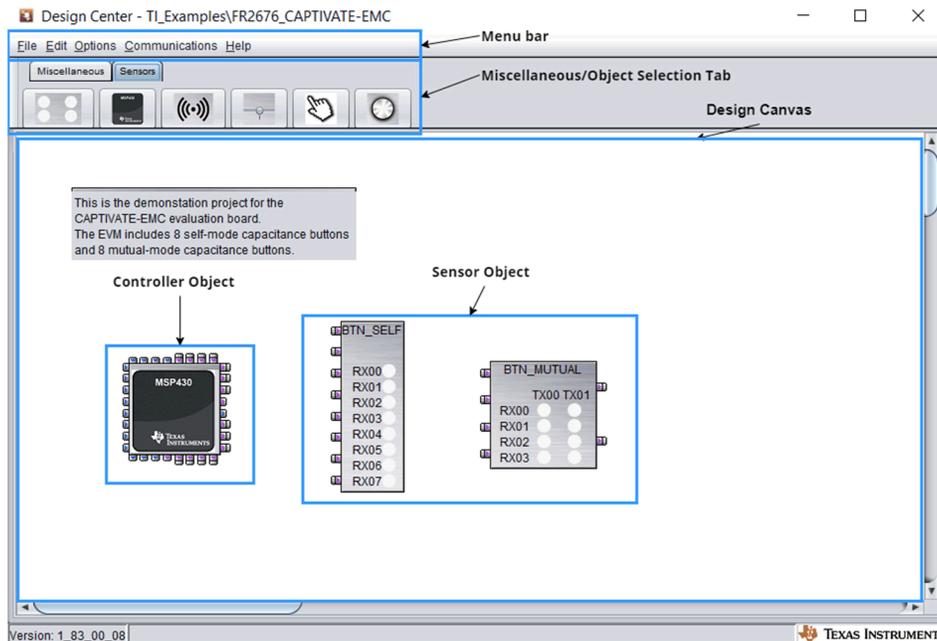


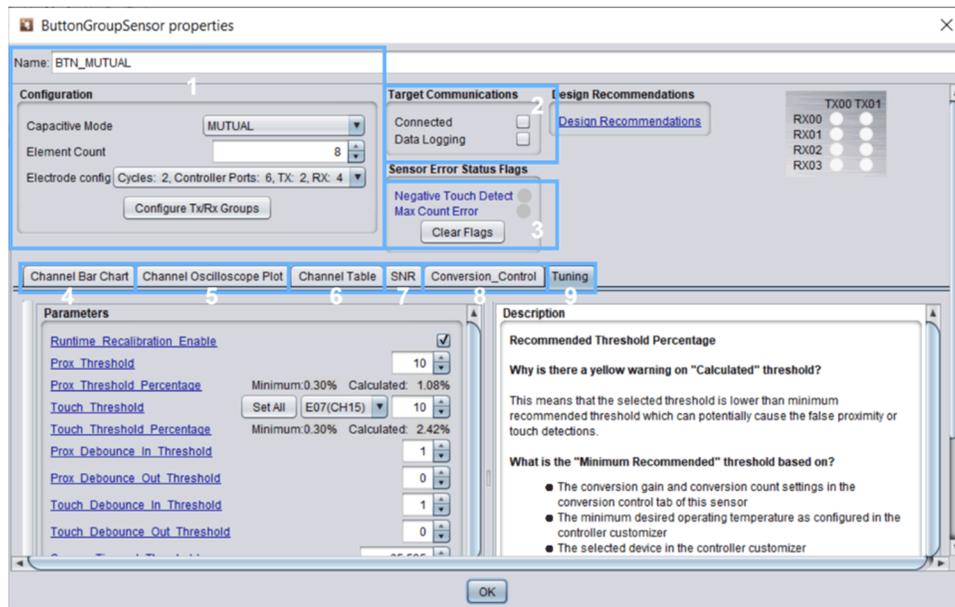
Figure 5-7. GUI Main Interface

Table 5-1. Parameters to Set in the GUI Main Interface

Reference	Operation	Function	Recommended Value	Comment
Menu bar	File	Create a new GUI project, or import examples to modify	None	
	Options	Click "Features" and select "Advanced" to enter the advanced tuning mode	"Advanced" mode	
	Help	"Topic" button can directly open the manual of capacitive touch	None	
Object Selection	Sensors	Drag the MCU widget and the required sensor widget to the design canvas	None	

5.2.2 Sensor Widget Configuration

MCU widget and sensor widget are the two most important parts of the GUI, covering functions such as code generation, online parameter adjustment, and sensor signal monitoring. First, we introduce the configuration required for the sensor widget, as shown in [Figure 5-8](#). This part mainly configures the name, quantity and mode of the sensor combination, and corresponds the virtual buttons to virtual pins. The parameters to be configured are shown in [Table 5-2](#). The numbers in the boxes correspond to those in [Figure 5-8](#). Other parameters are involved in Phase 2.


Figure 5-8. Sensor Widget Interface
Table 5-2. Parameters to Set in the Sensor Widget

Number	Operation	Function	Recommended Value
1	Name	Configure the name of the sensor combination Sensor, such as Button-pad.	Default values
	Capacitive Mode	Choose whether the Sensor belongs to mutual-capacitive capacitance mode or self-capacitive capacitance mode.	None
	Element Count	Used to select the number of Element to which the Sensor belongs, that is, how many buttons it occupies.	None
	Electrode config	In the mutual-capacitive mode, select the corresponding TX and RX according to the actual number of rows and columns designed.	None
	Configure Tx/Rx Groups	Used to understand the TX and RX occupied by each virtual Element, and map virtual buttons to virtual pins.	None

5.2.3 MCU Widget Configuration

The MCU widget interface is shown in Figure 5-9. This part mainly configures the MCU model (No. 1); MCU working mode and low power consumption parameters (No. 2, 5, 8), anti-noise function (No. 9); mapping virtual pins to actual pins (No. 4). The parameters that need to be configured are shown in Table 5-3. The numbers in the boxes correspond to those in Figure 5-9. Other parameters are involved in Phase 2.

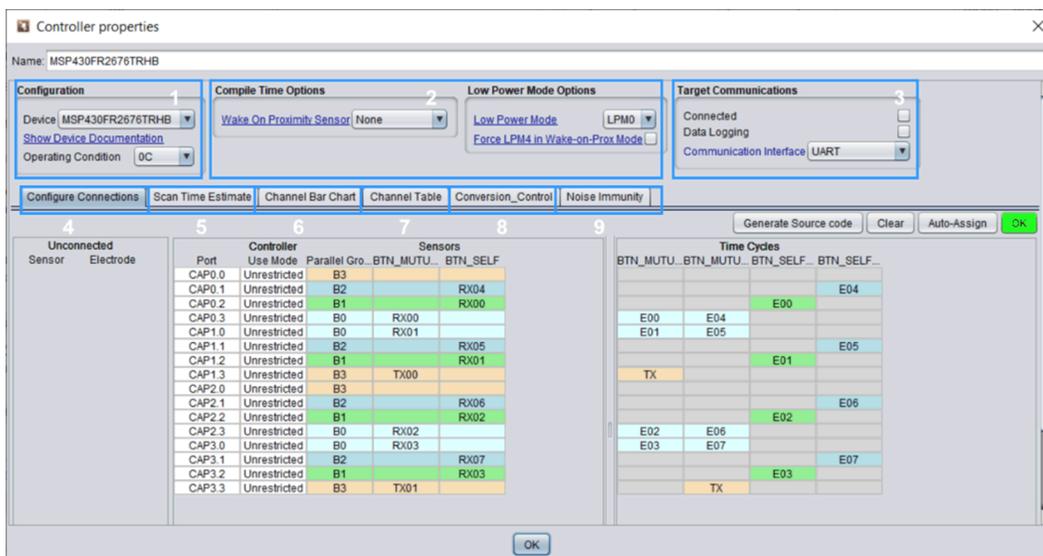


Figure 5-9. MCU Widget Interface

Table 5-3. Parameters to Set in the MCU Widget

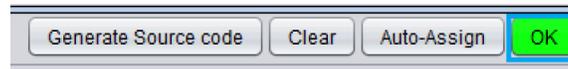
Number	Operation	Function	Recommended Value	Comment
1	Device	Select the device and package.	None	
	Operating Condition	Choose the working temperature.	25°C	Temperature affects the noise of the device and the Threshold Percentage in the sensor widget.
2	Wake on Proximity Sensor	If there is a need for ultra-low power consumption, choose a Sensor that is used as a proximity wake-up sensor.	Can be unenabled	
	Low-Power Mode	Choose the low-power mode according to system requirements.	Can use default configuration	LPM3 has lower power consumption than LPM0, but it cannot actively receive UART data and can only send data.
	Force LPM4 in Wake-on-Prox Mode	When enabled, the MCU consumes less power in Wake-on-Prox mode.	Can be unenabled	After enabling, the timing clock source is VLO clock, the timing error is relatively large.
3	Communication Interface	Choose UART or BUCK_I2C to communicate with the GUI.	None	See Section 4.2.1 for the communication pins and IRQ PIN used.
4	Configure Connections	According to the actual button design, map the virtual Tx and Rx pins to the CAP I/O pins.	Can click "Auto-Assign"	In addition to the button and proximity sensor function, it is recommended to select the Auto-Assign function.
5	Scan Time Estimate	Automatically calculate the total scanning time of the button in Active mode.	None	Used to evaluate the shortest "Active Mode Scan Rate ms".

Table 5-3. Parameters to Set in the MCU Widget (continued)

Number	Operation	Function	Recommended Value	Comment
8	Active Mode Scan Rate ms	Set the scan time interval of the sensor in active mode according to requirements.	Can use default configuration	
	Wake on Prox Mode Scan Rate ms	Set the scan time interval of the proximity sensor in Wake-on-Prox mode.	Can use default configuration	The shorter the time, the faster the scanning speed close to wake-up, but the higher the power consumption.
9	Enable Noise Immunity	Enable noise immunity function.	Can be unenabled	It is used to improve the anti-noise ability, but the amount of code is larger and the scanning time increases.
	Self/Mutual mode Oversampling	Adjust oversampling level.	Can be unadjusted	Can improve the level of noise immunity, but oversampling increases the scan time of a single cycle.
	Dynamic Threshold Adjustment	Enable dynamic adjustment of proximity sensing and touch threshold.	Enable	After enabling, the sensitivity changes with the noise level, but when encountering sudden noise, the threshold change may lag behind the noise change.

The "Configure Tx/Rx Groups" in the sensor widget store the Tx and Rx corresponding to the virtual button, and the "Configure Connections" in the MCU widget store the CAP X.X corresponding to the physical button. Correspond the physical buttons with the virtual buttons.

After configuration, as shown in [Figure 5-10](#), when the OK button in the "Configure Connections" directory turns green, and the virtual pins are mapped correctly to the actual pins, you can click "Generate Source Code" to generate the code. Remember creating a new project will **delete** all files in the current folder!


Figure 5-10. OK Button

5.3 Phase 2: Download Code

The download tool supports CCS and IAR. Refer to the [Loading and running generated projects](#) chapter in the CapTIvate Technology Guide for specific methods of downloading projects. If the import fails, look for an existing project with the same name under the CCS workspace. For IAR, download the latest version to support all capacitive-touch related MSP430 MCUs.

5.4 Phase 3: Adjust Parameters

The most important thing for capacitive touch is parameter setting. This section introduces all the commonly used parameters in detail.

5.4.1 Parameter Adjustment Logic and Method

The steps involved in tuning can be roughly divided into four parts: Data monitoring, sensitivity parameters adjustment, system reliability adjustment, response speed and power consumption adjustment. The overall idea is to first adjust the sensitivity, then adjust the reliability of the touch, and finally adjust the response time and power consumption. Remember to use the data monitoring function to evaluate the effect of parameter adjustment during the entire parameter adjustment process.

The logic block diagram of parameter adjustment is shown in [Figure 5-11](#). Each parameter adjustment step often needs to be repeated. It should be noted that the difficulty of the entire parameter tuning has a great relationship with the initial mechanical and hardware design. A good mechanical and hardware design mean that the noise of the touch signal is stronger, and the system reliability parameter adjustment is simpler.

The two button nodes in the entire development process are: whether the signal change generated by the touch is higher than the minimum threshold, whether the system can pass the reliability test. The former characterizes the amount of signal generated by the touch. If the amount of signal is too small, the system may be affected by noise and cause false touches. The latter characterizes whether the system can work normally under certain conditions. Failure to meet any of the above conditions means that the machinery and hardware need to be modified.

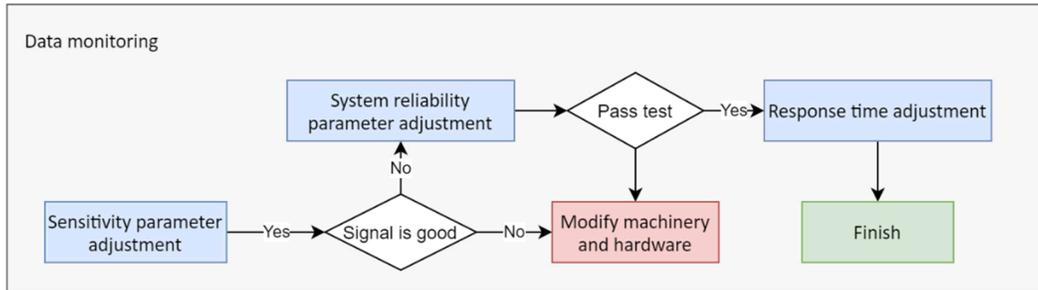


Figure 5-11. Logic of Parameter Adjustment

Before adjusting the parameters, first click Communications->Connect on the main GUI interface to connect the MCU to the GUI, and then adjust the relevant parameters according to the following subsections. In the process of online parameter adjustment, the following buttons in the sensor widget are used to control parameter update and reset, as shown in Figure 5-12. The corresponding functions are shown in Table 5-4. If there is no response from the GUI, remove and restore power and confirm that the HID Bridge UART/I2C is connected correctly.



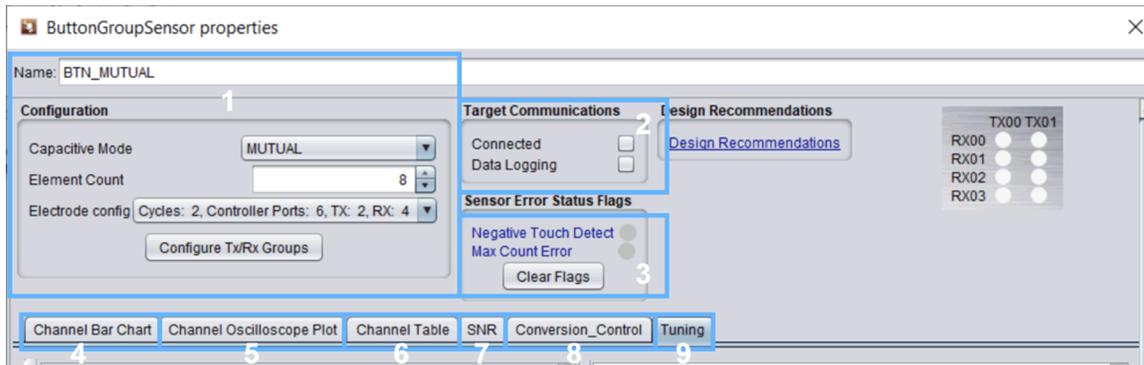
Figure 5-12. Online Tuning Button

Table 5-4. Buttons for Online Tuning

Number	Button	Description
1	Force Recalibration	Mandatory system calibration, used when the button function is not normal.
2	Apply	Transfer the parameters on the GUI to the MCU.
3	Read	Read the current relevant parameters in the MCU.
4	Reset	Reset the touch parameters in the GUI and MCU to the initial configuration values of the GUI.

5.4.2 Data Monitoring

The data monitoring function of the sensor is distributed in the MCU widget and the sensor widget at the same time. The difference between the two is that all the data of Sensor and Element can be observed in the MCU widget, while the sensor widget only displays the element data contained in it. Because the actual parameter adjustment is mainly for the sensor widget, this section only discusses the relevant functions in the sensor widget. The functions involved are shown in Table 5-5, and the numbers in the selection boxes correspond to those in Figure 5-13.


Figure 5-13. Sensor Widget Interface During Data Monitoring
Table 5-5. Sensor Data Monitoring

Number	Operation	Function	Comment
2	Data Logging	After enabling, save the real-time data collected by the button in the creation directory of the GUI project.	Generally used to evaluate the reliability of the system after recording a piece of data.
4	Channel Bar Chart	Observe the signal changes of each Element channel.	It can visually observe the relationship between the signal and the touch threshold and the proximity sensor threshold. It is mainly used for sensitivity adjustment.
5	Channel Oscilloscope Plot	Show the relationship between data and time of each Element channel.	Observe the signal changes within a period of time, mainly used for system reliability adjustment.
6	Channel Table	Observe the real-time signal changes of each Element channel through a table.	The data can be directly observed, which is generally used to evaluate the rationality of the threshold setting and whether there is enough margin.
7	SNR	Multiple sampling to evaluate the rationality of the entire system design and threshold setting.	

5.4.3 Sensitivity Parameter Adjustment

The adjustment of the sensitivity parameter is mainly based on the actual detection requirements to select the appropriate system gain and the appropriate threshold. Increasing the gain of the CapTIvate system infinitely amplifies the noise while amplifying the touch signal. In the end, it only increases the resolution of the system, but the signal-to-noise ratio itself does not change. Because a larger gain increases the conversion time, it is not recommended to adjust the gain of the entire system too high. The functions involved in this part are shown in [Table 5-6](#), and the numbers in the selection boxes correspond to those in [Figure 5-13](#).

Table 5-6. Sensitivity Parameters

Number	Operation	Function	Recommended Value	Comment
8	Conversion Count	Set the equivalent setting value of the capacitance corresponding to the Sensor.	<1000	Determine the system resolution.
	Conversion Gain	Set the system gain corresponding to the Sensor.	100	The minimum value is 100.
	Frequency Divider	Set the frequency division coefficient of charge sampling and conversion.	Can use default value	If the parasitic capacitance is too large, you need to grab the waveform to check whether the frequency division coefficient needs to be adjusted.

Table 5-6. Sensitivity Parameters (continued)

Number	Operation	Function	Recommended Value	Comment
9	Prox Threshold	Set proximity sensor threshold.	Make Prox Threshold Percentage greater than the Minimum value	The absolute change threshold of the touch signal, all elements share a proximity sensing threshold.
	Touch Threshold	Set touch threshold.	Ensure priority triggering of proximity sensing threshold	The threshold of the percentage change of the touch signal. At the same time, adjust to select " select all ", all elements have their own touch thresholds.
	Desired Resolution	The resolution of the slider, scroll wheel or touch panel.	<100	Excessive resolution is easily affected by noise.

5.4.4 System Reliability Parameter Adjustment

The adjustment of system reliability parameters mainly reduces the influence of noise from the perspective of software and handles abnormal modes. The functions involved in this part are shown in [Table 5-7](#), and the numbers in the selection boxes correspond to those in [Figure 5-13](#).

Table 5-7. System Reliability Parameters

Number	Operation	Function	Recommended Value	Comment
8	Modulation Enable	Set frequency jitter function.	Can be disable	For the self-capacitive mode, the frequency jittering function is enabled, and the mutual-capacitive is not enabled.
9	Pro/Touch Debounce In Threshold	Set the delay time for the system to determine the occurrence of proximity sensing or touch events.	1-3	Response delay is in Active Mode Scan Rate ms.
	Pro/Touch Debounce out Threshold	Set the delay time for the system to judge that there is no proximity sensor or touch event.	1-3	Response delay is in Active Mode Scan Rate ms.
	Sensor Timeout Threshold	Set the time for the system to reset calibration when the proximity sensor is triggered continuously.	Set according to time requirements	The time is in Active Mode Scan Rate ms, which can prevent continuous forward signal drift caused by environmental changes.
	Negative Touch Threshold	Set the threshold for the negative change of the equivalent "Count" value of the capacitor.	Default value	The threshold value of the absolute change of the touch signal can prevent the negative signal change problem caused by touch power-on.
	Counter Filter Beta	Set the strength of IIR filtering for the equivalent "Count" value of the capacitor.	1-3	Too much assembly affects the response speed.
	LTA Filter Beta	Set the strength of IIR filtering for LTA.	7	Changing this size can cause unresponsiveness after multiple touches.
Error Threshold	Set the maximum value of the charge conversion output.	conversion count × 1.25	Increase the processing logic according to the flag bit bMaxCountError.	

Noise sources include the CapTIvate module itself, digital signals in the MCU, the power supply, and the input signal at the capacitive touch pin. Reasonable parameters can reduce the impact of noise, but they often need to be adjusted repeatedly. The objects of repeated adjustments are mainly "Pro/Touch Debounce In Threshold" and "Counter Filter Beta" functions. At the same time, make sure that the noise Immunity function in the MCU widget is enabled.

5.4.5 Response Speed and Power Consumption Adjustment

One that affects the response speed is the scan time, and the other is the hysteresis effect of the touch signal. The functions involved in this part are shown in [Table 5-8](#).

Table 5-8. Response Speed Adjustment Parameters

Widget	Subdirectory	Function	Comment
MCU widget	Conversion_Control	Active Mode Scan Rate ms	Scan period setting of the sensor.
	Noise Immunity	Enable	Affect the actual scan time.
Sensor widget	Conversion_Control	Conversion Count	Too large a parameter means that the number of charge transfers increases, which affects the actual scan time.
	Tuning	Debounce In/Out Threshold	Affect the speed of signal change.
		Counter Filter Beta	Affect the speed of signal change.

In addition, it is not recommended that the host use the I2C fixed cycle polling method to obtain the button status information of the MSP430, because this must ensure that the MSP430 sensor scan speed is twice the polling speed of the host, which indirectly reduces the upper limit of the button response speed of the entire system.

The longer the scan time, the faster the scan frequency means the higher the power consumption. It is recommended to use the Energytrace function provided by [CAPTIVATE-PGMR](#) to optimize the system by observing the relationship between time and power consumption. Note that the power supply pin used is **3V3 Metered**. If you have higher requirements for power consumption, it is recommended to enable LPM3 mode and Wake-on-Prox mode.

EnergyTrace: A code analysis tool for optimizing power consumption, which can be used to measure and display the power consumption of an application, including software and hardware. The software part is integrated in CCS and IAR.

- Home page: [EnergyTrace Technology](#)
- User guide: [ULP Advisor™ Software and EnergyTrace™ Technology](#)

5.5 Phase 4: Modify Communication Mode

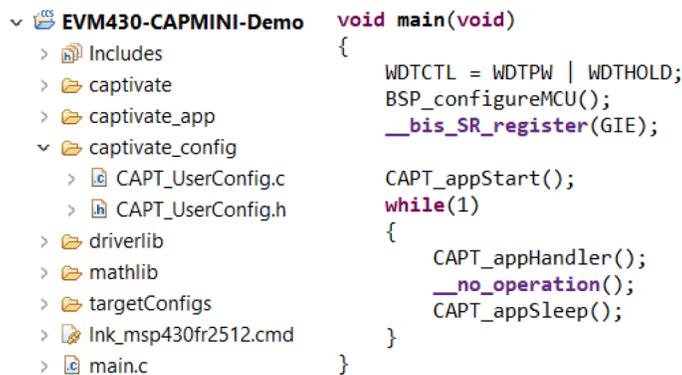
After debugging the capacitive touch parameters, the next step is to use other host computers to try to establish a connection with the MSP430. Communication influence suggests to select "None", and then use the GUI to regenerate the code, or directly modify `captivate_config->CAPT_UserConfig.h->` the macro definition of `CAPT_INTERFACE`.

5.6 Phase 5: Develop Custom Applications

Developing custom applications means understanding the code composition of the CapTlvate peripheral. The following sections describe the most common custom requirements.

5.6.1 Program Structure

Figure 5-14 shows the composition of the program. For more related instructions, refer to [Starting from Scratch with the Starter Project](#).



```

void main(void)
{
    WDTCTL = WDTPW | WDTHOLD;
    BSP_configureMCU();
    __bis_SR_register(GIE);

    CAPT_appStart();
    while(1)
    {
        CAPT_appHandler();
        __no_operation();
        CAPT_appSleep();
    }
}
    
```

Figure 5-14. Demo Program Folder Structure

The directory structure of the software project is shown on the left side of [Figure 5-14](#). The folders or files involved in this part are shown in [Table 5-9](#). Among them, the most core for users is the CapTlvate_config folder. As shown in [Figure 5-15](#), the corresponding relationship between the GUI and the parameters in CapTlvate_config can be clicked on the corresponding parameter in the GUI and read "Affected Software Parameters" for understanding.

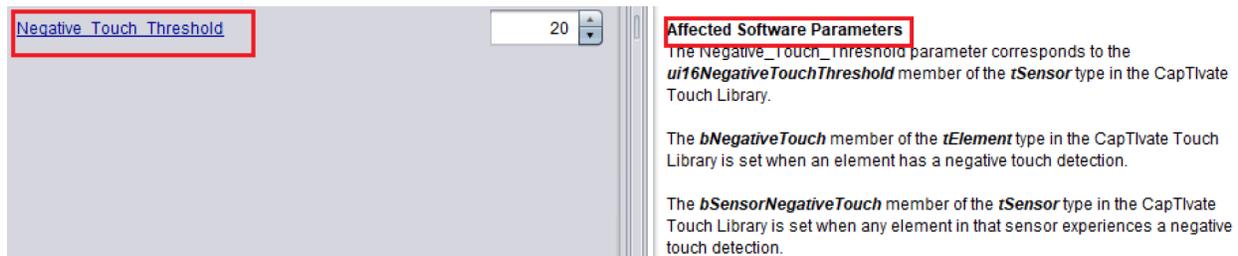


Figure 5-15. Correspondence Between GUI and Code Parameters

Table 5-9. Tools for Software Development

Number	Contents	Comment
1	CapTlvate	Low-level driver library for capacitive touch-related peripherals, including capacitance detection and communication configuration.
2	CapTlvate_app	The top-level logic code of capacitive touch, including system initialization, scanning and calibration logic.
3	CapTlvate_config	Configuration parameters generated by the GUI.
4	driverlib	MSP430 peripheral driver library.
5	mathlib	Fixed-point calculation library.
6	targetConfigs	MCU model selection and programming configuration (no need to change).
7	Ink_msp430fRxxxx.cmd	Linker command file, which gives the settings of program space and data space.
8	main.c	Main function

The program structure of the main function is shown on the right side of [Figure 5-14](#), and the function functions involved in this part are shown in [Table 5-10](#). For more function descriptions, refer to the [API Guide](#).

Table 5-10. Important Functions

Number	Contents	Comment
1	WDTCTL = WDTPW WDTHOLD	Ban watchdog.
2	BSP_configureMCU()	Configure the communication IO pin, configure the clock.
3	__bis_SR_register(GIE)	Enable global interrupt.
4	CAPT_appStart()	Capacitive touch module initialization and calibration, while enabling communication peripherals.
5	CAPT_appHandler()	Capacitive touch scanning and mode switching.
6	__no_operation()	CPU waits for one cycle.
7	CAPT_appSleep()	CPU enters sleep, waiting for periodic interrupt to wake up.
8	CAPT_initUI()	Capacitive touch module initialization, and enable communication peripherals.
9	CAPT_calibrateUI()	Capacitive touch module calibration.
10	CAPT_updateUI()	Update the parameters of all sensors.
11	I2CSlave_setRequestFlag()	Used to generate high-level pulses when touched to remind the host computer of touch events.

5.6.2 Capacitive Touch Status and Parameter Reading and Processing

As described in [Section 5.1.3](#), the Sensor structure is an abstraction of the actual sensor module. The Cycle structure is under the Sensor structure, and the Element structure is under the Cycle structure. Users can directly access the structure declared in `caplvate_config->CAPT_UserConfig.c` to know the status of each module, and can also modify the parameters in the structure in real time according to their needs. The most

commonly used structures are Sensor and Element. Sensor can be directly accessed globally, and the Element structure needs to be declared as a global variable and then accessed.

The most commonly used variables of the Sensor structure are shown in [Table 5-11](#).

Table 5-11. Commonly Used Variables in the Sensor Structure

Number	Variable	Type	Comment
1	.bSensorProx	Bool	Indicate the Element under the Sensor whether detect a proximity event.
2	.bSensorTouch	Bool	Indicate the Element under the Sensor whether detect a touch event.
3	.bSensorPrevTouch	Bool	Indicate the Element under the Sensor whether detect a touch event in the previous scan cycle.
4	.pSensorParams->SliderPosition.ui16Natural	uint16_t	Get the current wheel or slider position.
5	.pvCallback	void (* pvCallback)(struct tSensor *)	The callback function can be linked to the sensor to process the state change of the Sensor during scanning, and it is called every time the button is scanned.

The most commonly used variables of the Element structure are shown in [Table 5-12](#).

Table 5-12. Commonly Used Variables in the Element Structure

Number	Variable	Type	Comment
1	.bProx	Bool	Indicates if the Element has detected a proximity sensor.
2	.bTouch	Bool	Indicates if the Element detect a touch.

The state change of the processing button can be realized through the callback function of the Sensor. For specific instructions, refer to [Register a callback function](#) ★. For examples of custom projects, refer to the CCS/IAR routines corresponding to EVM430-CAPMINI: [EVM430-CAPMINI-Demo](#) ★.

5.6.3 Customization of Communication Functions

It is recommended to use the provided UART and I2C library functions for development to communicate with the host MCU. First of all, the default code generated by the GUI completes the I/O configuration and clock configuration of UART (eUSCI_A0) and I2C (eUSCI_B0) in the BSP_configureMCU() function. Therefore, it is only necessary to configure the communication module and write the communication protocol. If you need to modify the communication interface, check the datasheet to see if you need to modify the SYSCFG3 register to redirect the communication port.

UART-based communication development:

- Communication Interface configuration: NONE
- Clock configuration: BSP_configureMCU()
- IO configuration: BSP_configureMCU()
- UART peripheral configuration and interrupt usage: [Driverlib UART loopback](#)
 - Comment :
 - Baud rate parameter configuration tool: [MSP430 USCI/EUSCI UART Baud Rate Calculation](#)
 - See the frequency of different system clock sources at `capivate_app->CAPT_BSP.h`

There are two schemes for I2C-based communication development for reference. The scheme occupies less code, and the second scheme saves development time.

Option one:

- Communication Interface configuration: NONE
- Clock configuration: BSP_configureMCU()
- IO configuration: BSP_configureMCU()
- I2C peripheral configuration and interrupt usage: [Driverlib I2C salveTxMultiple](#), [Driverlib I2C salveRxMultiple](#)

Option two★:

- User Guide : [I2C Slave driver user guide](#)
- Communication Interface configuration: NONE
- Clock configuration: BSP_configureMCU()
- IO configuration: BSP_configureMCU()
- I2C peripheral configuration and interrupt usage: The code under the REGISTER_I2C configuration includes I2C peripheral configuration and interrupt usage. The user only needs to analyze the 3-byte transmit and receive Buffer. The operation steps are as follows:

1. Modify the CAPT_I2CRegisterReceiveHandler() function in captivate->COMM->CAPT_Interface.c. This function is called when the stop signal or restart signal are generated after the I2C write operation is completed. As shown in [Figure 5-16](#), delete other codes and add custom frame processing functions.

```
static uint8_t g_ui8I2CDataBuffer[CAPT_I2C_REGISTER_RW_BUFFER_SIZE];
static bool CAPT_I2CRegisterReceiveHandler(uint16_t ui16Cnt)
{
    Cus_CAPT_CommunicationPackageAnalysis(g_ui8I2CDataBuffer);
    //
    // Return false to exit with the CPU in its previous state.
    // Since the response packet generation was handled immediately here,
    // there is no need to exit active.
    //
    return false;
}
```

Figure 5-16. CAPT_I2CRegisterReceiveHandler() Function Modification

2. Write a custom frame processing function. As shown in [Figure 5-17](#), the MSP430 MCU reads a byte sent by the host, then modify the first byte of the data segment in the Buffer, and then wait for the host to actively read the byte. It should be noted that I2C sending and receiving share a 32-byte buffer, and the whole process does not reset it. In addition, the receiving/sending data segment in the buffer has an offset of 3 bytes. Therefore, the fourth byte of the Buffer corresponds to the first byte of the received/sent data.

```
void Cus_CAPT_CommunicationPackageAnalysis(uint8_t * pBuffer)
{
    switch (pBuffer[3])
    {
        case CUSTOM_KEY_STATE_CMD:
            pBuffer[3] = keyValue;
            break;
        case CUSTOM_DEVICE_NUMBER_CMD:
            pBuffer[3] = DEVICE_NUMBER;
            break;
        default:
            break;
    }
}
```

Figure 5-17. Frame Processing Function

3. Build the overall code as shown in [Figure 5-18](#). The additional functions included are:
 - Panel_Init(): System custom initialization, register the sensor callback function.
 - BTNEventHandler(): The processing function that defines the state of the sensor.

```

50#include <msp430.h>
51#include "driverlib.h"
52#include "captivate.h"
53#include "CAPT_App.h"
54#include "CAPT_BSP.h"
55#include "applications.h"
56#include "I2CSlave.h"
57
58#define CUSTOM_KEY_STATE_CMD 0x0B //I2C CMD Byte
59
60void BTNEventHandler(tSensor *pButton)
61{
62    if((pButton->bSensorTouch == 1) &&
63        (pButton->bSensorPrevTouch == 0))
64    {
65        I2CSlave_setRequestFlag();
66    }
67}
68
69void Cus_CAPT_CommunicationPackageAnalysis(uint8_t * pBuffer)
70{
71    switch (pBuffer[3])
72    {
73        case CUSTOM_KEY_STATE_CMD:
74            pBuffer[3] = BTN00.bSensorTouch;
75            break;
76        default:
77            break;
78    }
79}
80void Panel_Init()
81{
82    //Customized initialization
83    BTN00.pvCallback = (pvCallbackAddress)&BTNEventHandler;
84
85}
86
87void main(void)
88{
89    WDTCTL = WDTPW | WDTHOLD;
90    BSP_configureMCU();
91    Panel_Init();
92    CAPT_appStart();
93
94    while(1)
95    {
96        CAPT_appHandler();
97        CAPT_appSleep();
98    }
99}

```

Figure 5-18. Diagram of the Overall Code

Table 5-13 shows the information related to the custom communication function.

Table 5-13. Customized Communication

Number	Object	Modification method	Comment
1	eUSCI_A0/eUSCI_B0 port	Modify the pin definition in Captivate_app->CAPT_BSP.C->BSP_configureMCU()	Select other communication peripherals, need to modify the initialization of the peripherals.
2	IRQ signal of I2C	Modify the macro definition of IRQ pin in Captivate->COMM->CAPT_CommConfig.h Modify the macro definition of IRQ pin action in Captivate->COMM->I2CSlave.h	This pin is used to remind the host computer of a touch event.
3	Whether the CapTlvate module enables the Timer peripheral	Modify the assignment of FUNCTIONTIMER__ENABLE in CapTlvate->COMM->FunctionTimer.h	Used for timeout function in I2C communication.
5	Timeout function of I2C communication	Modify the assignment of I2CSLAVE__TIMEOUT_ENABLE in Captivate->COMM->I2CSlave_Definitions.h Modify the assignment of I2CSLAVE__TXFR_TIMEOUT_CYCLES in Captivate->COMM->I2CSlave_Definitions.h	By default, the host resets I2C if a single frame transmission exceeds 8 ms.
5	Modify I2C address	Modify I2CSLAVE__ADDRES in CapTlvate->COMM->I2CSlave_Definitions.h	

5.6.4 Bootloader

Bootloader is mainly used for online upgrade. The hardware needs to occupy 4 pins. Two pins (RST, TEST) are used to select the bootloader mode, and the remaining 2 pins are used for communication. Because the capacitive touch function needs to occupy a lot of code space, it is recommended to use the ROM Bootloader that comes with the MSP430 MCU.

User Guide:

[MSP430 FRAM Devices Bootloader \(BSL\) User's Guide](#)

Application Guides:

[MSP430 Bootloader With SimpleLink MCUs](#)

[MSP430 Bootloader With Sitara Embedded Linux Host](#)

5.6.5 Test, Production, and Programming

A successfully designed capacitive touch sensing system requires the use of prototype equipment for system verification. TI recommends building 20 to 50 devices for field testing. If the system meets the performance requirements, the design can shift to mass production. If the system does not meet the performance requirements, it is necessary to adjust the performance expectations or redesign the software and hardware, or even the mechanical structure design.

Due to the uncertainty of the human body's capacitance to ground, which is different from the thickness of different hands, it is not recommended to use human hands for related tests. It is recommended to use grounding copper pillars of different thicknesses to simulate the touch of a human hand. A thick copper column is used to characterize the lower limit of sensitivity. At this time, each test button can correctly respond to touch signals. Use thin copper pillars to characterize the lower limit of sensitivity. At this time, each test button should not respond to touch signals. The thickness of the specific copper pillars needs to be designed according to the actual sensitivity requirements.

In small batch production, the recommended software is UniFlash and the recommended tool is eZ-FET or MSP-FET.

UniFlash★: UniFlash is a programming GUI tool developed by TI, which supports JTAG and BSL. To program MSP430, You need to load a binary file, which can be generated as described in [this post](#).

- Product page: [UniFlash](#)
- User's guides:
 - [UniFlash Quick Start Guide](#)
 - [Programming the Bootloader of MSP430™ Using UniFlash](#)

MSP-FET★: The most powerful and fastest MSP430 debug probe. Target VCC is selectable and the maximum supply current is 100 mA.

- Product page: [MSP-FET MSP MCU Programmer and Debugger](#)

eZ-FET★: A low-cost MSP430 debug probe and usually sold with LaunchPad kits. Supports a fixed voltage power supply.

- Product page: See the specific LaunchPad kit product page, for example, [MSP430FR2311 LaunchPad kit](#).

In mass production, MSP-GANG and its supporting upper computer software are recommended.

MSP-GANG★: The MSP Gang Programmer cannot debug code and is used for product production. It can be operated without a PC and supports programming eight MSP430 MCUs at the same time.

- Product page: [MSP-GANG Production Programmer](#)

6 Installation and Process Overview

CapTIvate Design Center is for code generation and online reference. The Code Composer Studio™ IDE (CCS) is for downloading GUI-generated code and embedded software development. The following sections describe how to install CCS and the CapTIvate Design Center step-by-step, how to quickly evaluate the EVM board provided by TI, and how to quickly develop your own applications system.

6.1 Installation of Code Composer Studio IDE

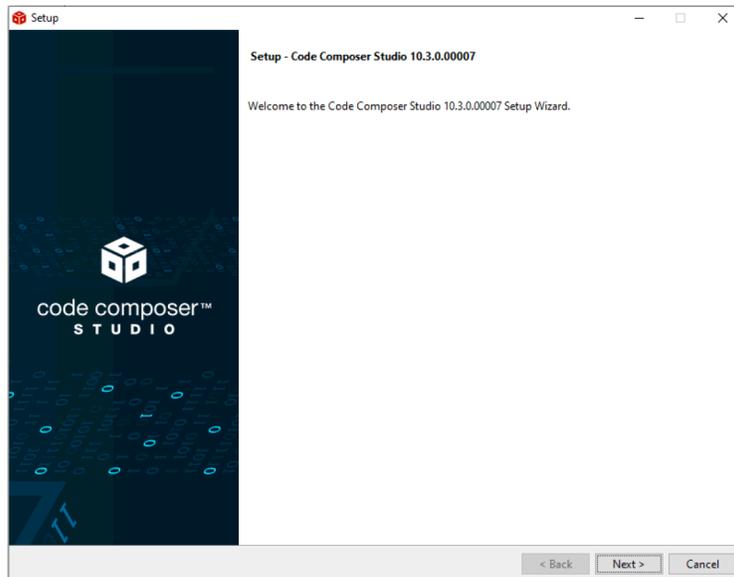
1. Download the Code Composer Studio IDE.

Product page: [CCS IDE for MSP430](#)

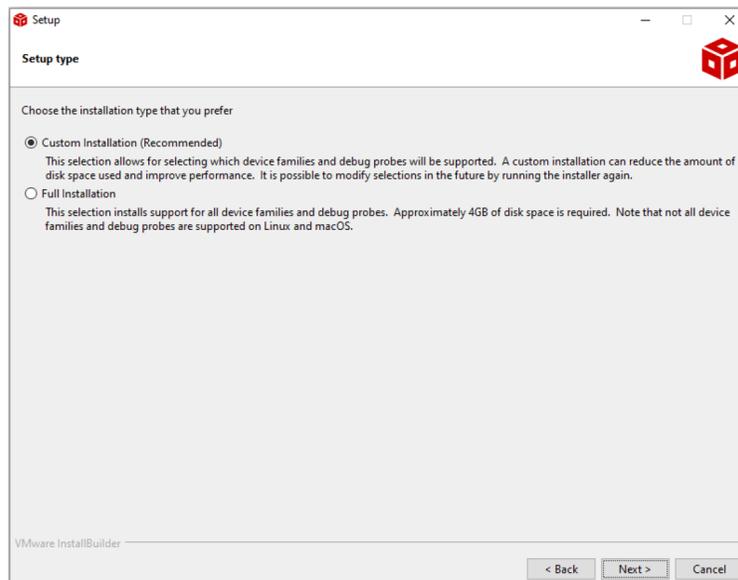
2. Select the single file installer or the on-demand installer for CCS IDE according to your PC operation system



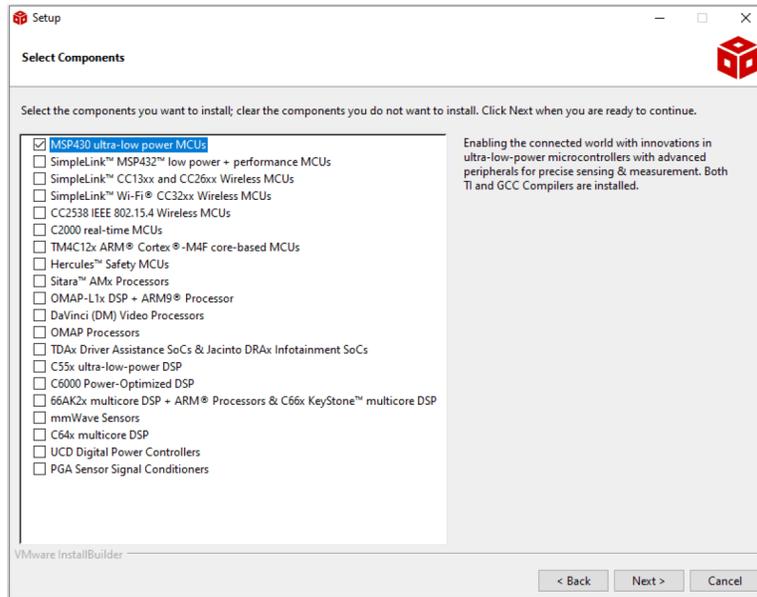
3. Start CCS IDE installation.



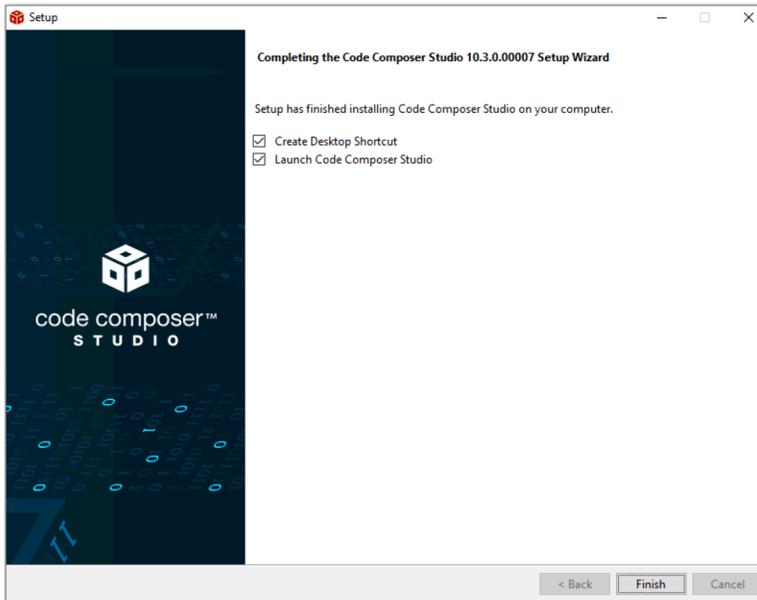
4. Select "Custom installation" is recommended.



- Select “MSP430 ultra-low power MCUs”.



- The installation is complete.



6.2 Installation of CapTIvate Design Center

Note

Install the Java JDK before installing the CapTIvate Design Center.

- Download CapTIvate Design Center.
Product page: [CapTIvate Design Center](#)



Get started

Step 1: Download and install the [CapTivate Design Center](#)

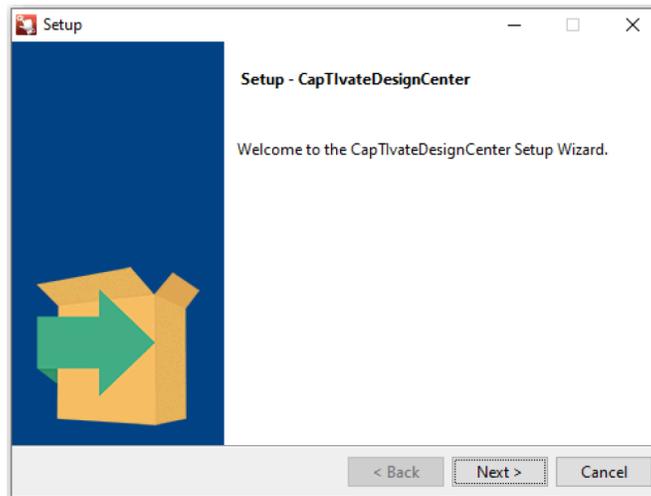
Step 2: Select and buy the best evaluation hardware for your [application](#)

Step 3: Complete the [getting started workshop](#)

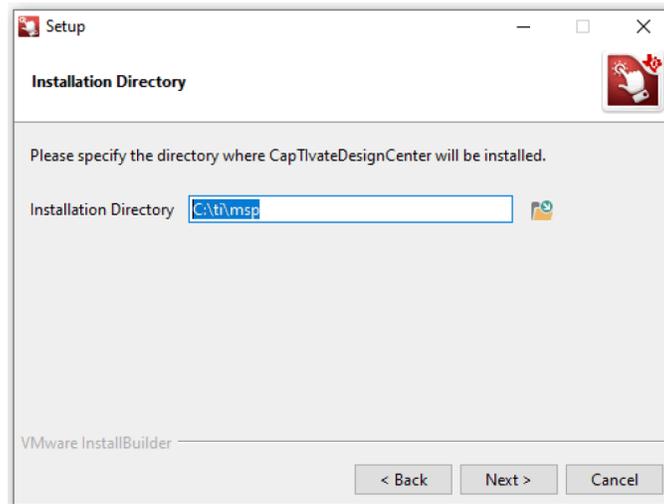
CapTivate_Design_Center Product downloads

Title	Description	Size
CapTivateDesignCenter-1.83.00.08-windows-installer.exe	Captivate Design Center for Windows	56784K
CapTivateDesignCenter-1.83.00.08-osx-installer.app.zip	Captivate Design Center for Mac OS X	53424K
CapTivateDesignCenter-1.83.00.08-linux-x64-installer.run	Captivate Design Center for Linux	56232K
md5sum.txt	MDS Sum	4K
Captivate Design Documentation		
Release_Notes_CapTIVateDesignCenter_1_83_00_08.html	Release Notes	80K
API Guide	API Guide	8K
Users Guide	Users Guide	4K

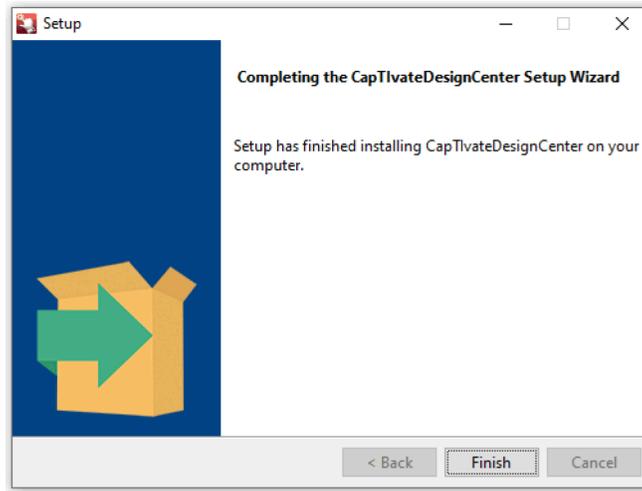
2. Start installation.



3. Keeping the default installation directory is recommended.



4. Installation is complete.



6.3 Hardware Connection

1. Rapid buildup the hardware evaluation platform using programmer, MCU board and sensor board.

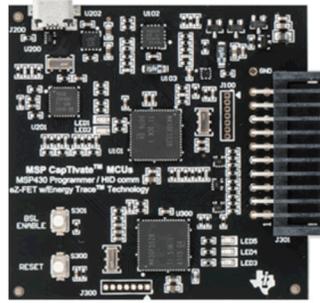
The hardware connection, rapid evaluation, rapid development demonstration are based on the development chain in the following table. For other EVM boards, the development and evaluation are similar.

GUI	IDE	SDK	Programmer	MCU Board	Sensor Board	Supported Sensor Type
CapTivate Design Center	CCS, IAR	MSPWare	CAPTIVATE-PGMR	CAPTIVATE-FR2676, CAPTIVATE-FR2633	CAPTIVATE-BSWP	Self-mode button, wheel, slider, proximity
					CAPTIVATE-PHONE	Mutual-mode button, wheel, slider, proximity
				BOOSTXL-CAPKEYPAD (MSP430FR2522)		12 mutual-mode buttons and 1 proximity
				EVM430-CAPMINI (MSP430FR2512)		4 self-mode buttons



2. CAPTIVATE-PGMR programmer

[Programmer product page](#)



CAPTIVATE-PGMR programmer interface definition

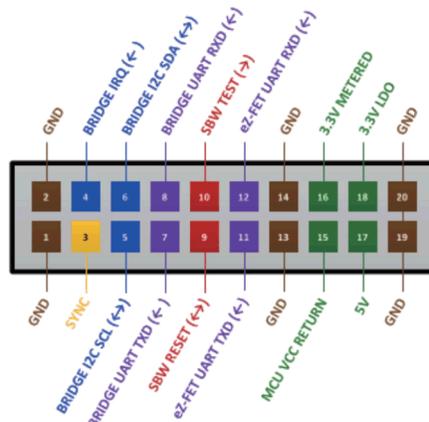
Power pins: 3.3V METERED and 3.3V LDO.

NOTE: METERED pin is used for EnergyTrace function, but, the power supply ripple noise is obvious. If EnergyTrace function is not needed, LDO pin should be used for power supply to reduce the ripple noise.

Programming pins: SBW TEST and SBW RESET

Debug pins: eZ-FET UART RXD and eZ-FET UART TXD. Notes: The function is the USB to serial port, which is different with the pins for online parameter tuning.

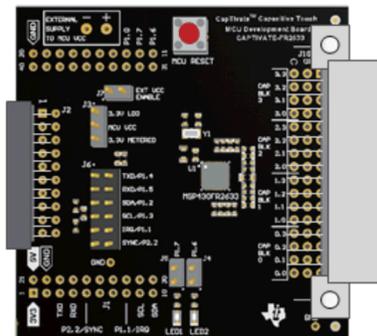
Captivate touch online parameter tuning pins: BRIDGE UART RXD and BRIDGE UART TXD, BRIDGE I2C SDA and BRIDGE I2C SCL.



PCB 20-PIN Male Connector

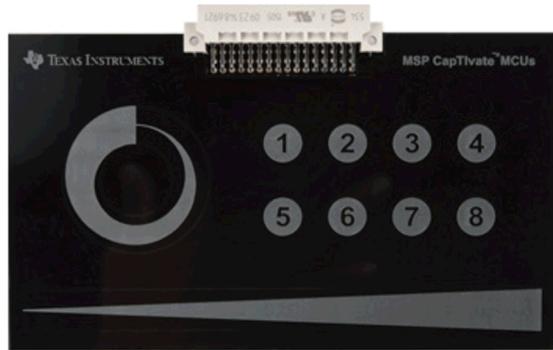
3. CAPTIVATE-FR2633 MCU board

[MCU board product page](#)



4. CAPTIVATE-BSWP Capacitive touch self-capacitance button, slider, wheel, and proximity sensor demonstration board

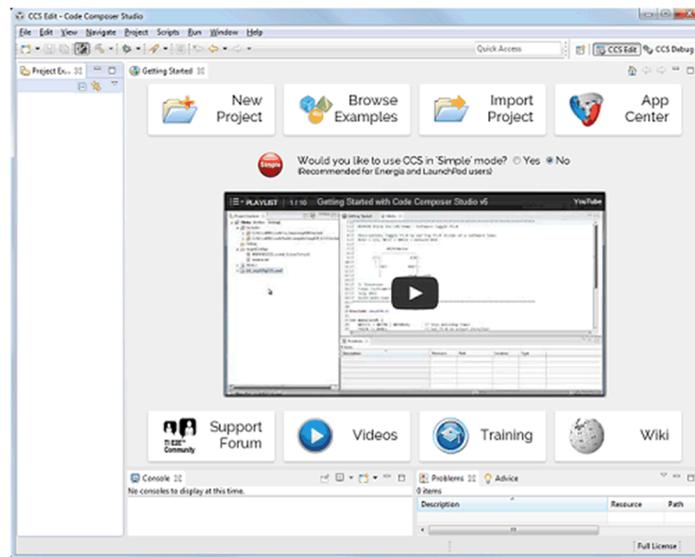
[Capacitive touch demonstration board product page](#)



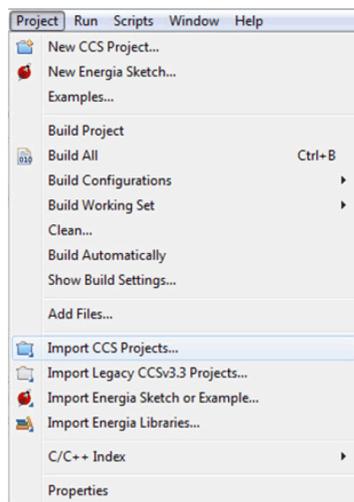
6.4 Rapid Evaluation

When CapTIvate Design Center is installed and the hardware validation platform is ready, import and open the example project for CapTIvate Design Center on CCS to rapidly evaluate the capacitive touch system.

1. Open CCS.



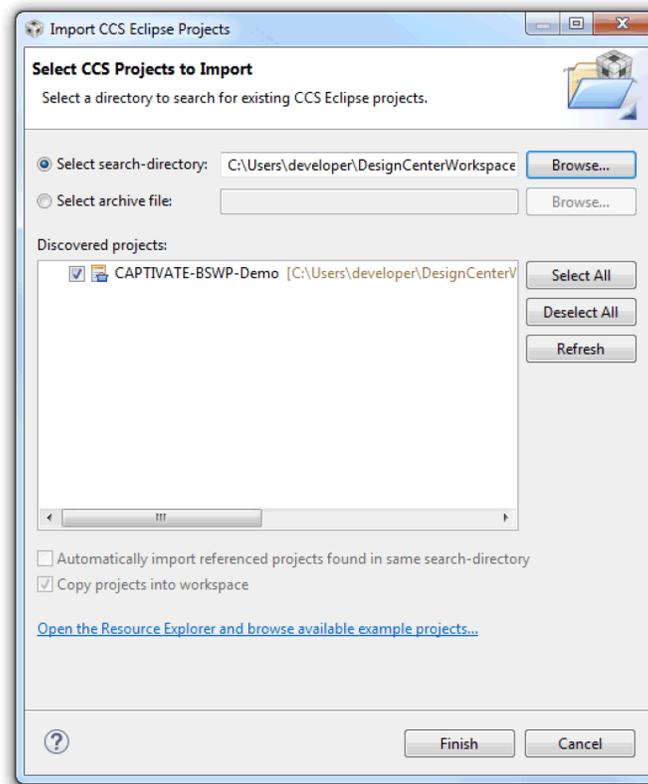
2. Import the CCS project.
Select Project→Import CCS Projects.



3. Select MSP provided capacitive touch example project on CapTIvate Design Center default folder.

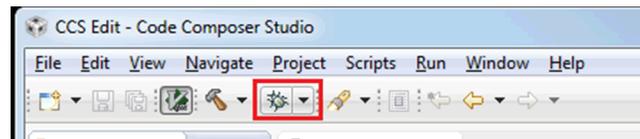
The default path is

C:\Users\Username\CapTlvateDesignCenter_x_xx_xx_xx\CapTlvateDesignCenterWorkspace\TI_Examples\FR2676_CAPTIVATE-BSWP



4. Tune and evaluate the capacitive touch example project.

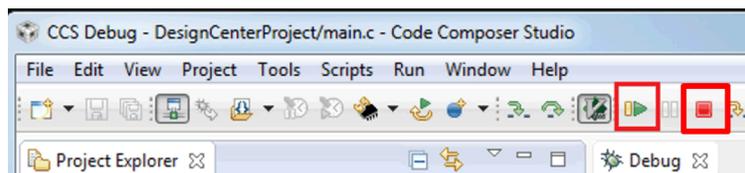
Click the debug button on CCS IDE.



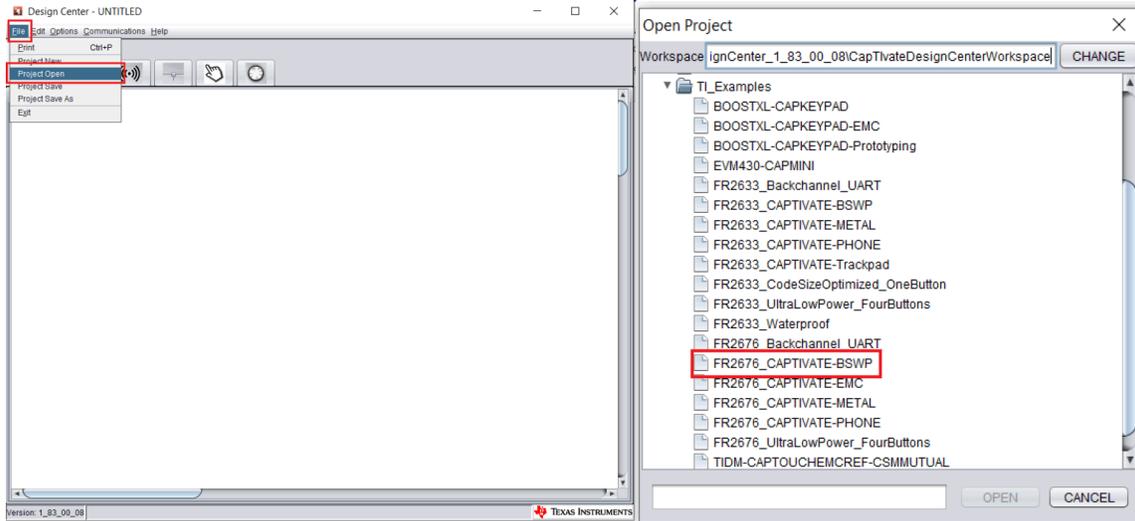
The first compile might take a long time.

warning #10366-D: automatic library build: using library "C:\ti\ccs1030\ccs\tools\compiler\ti-cgt-msp430_20.2.4.LTS\lib\rts430x_lc_ld_eabi.lib" for the first time, so it must be built. This may take a few minutes.

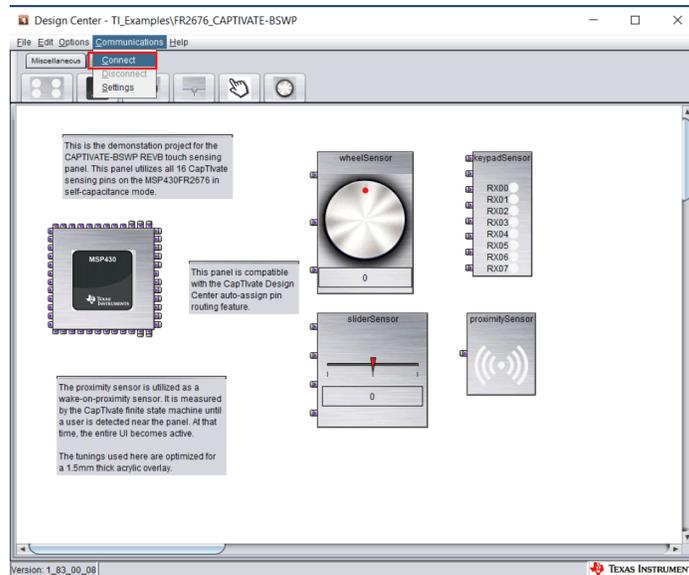
Click the run button and stop button. Then you may need to connect the power supply again.



5. Select and import the capacitive touch example project according to the capacitive touch demonstration board adopted.

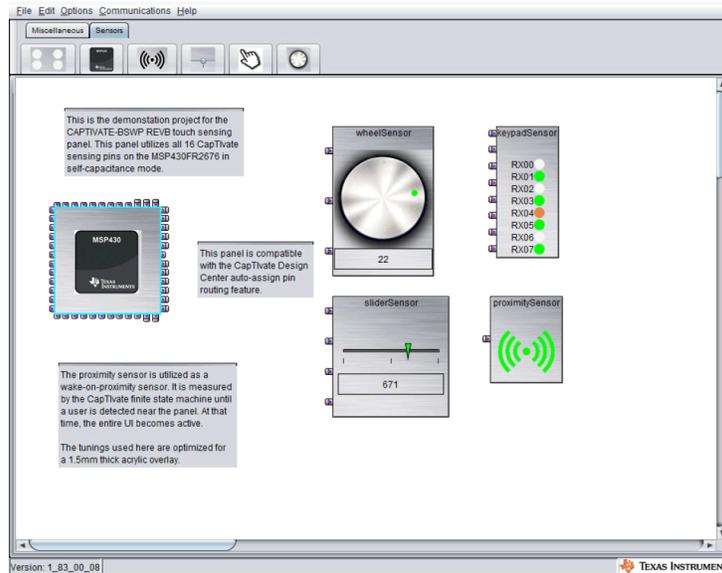


6. Connect CapTlvate Design Center with the hardware board system.



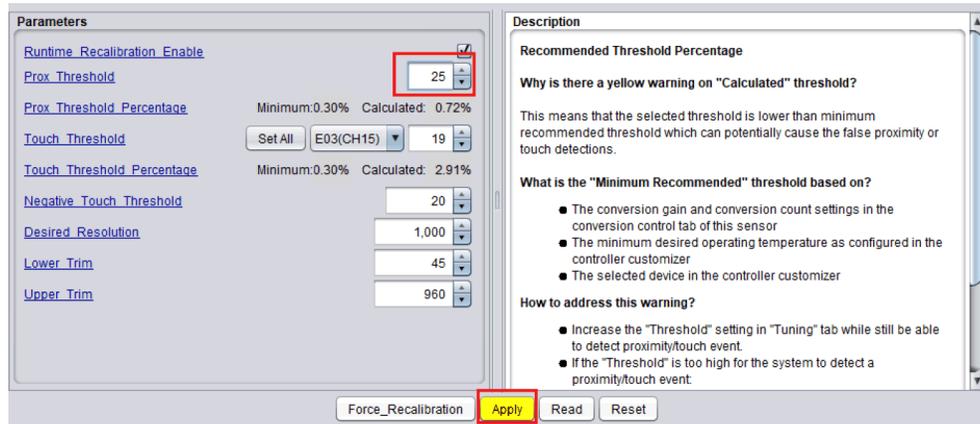
7. Evaluate the capacitive touch features.

White color indicates no signal triggering, yellow color indicates proximity sensing triggering, and green color indicates touch triggering.



8. Online parameter tuning

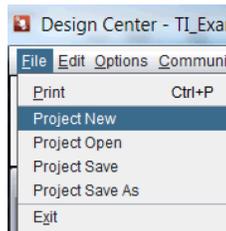
Modify the parameters and click the apply button to complete the online parameters tuning, see [Section 5.4](#) for further instructions and descriptions.



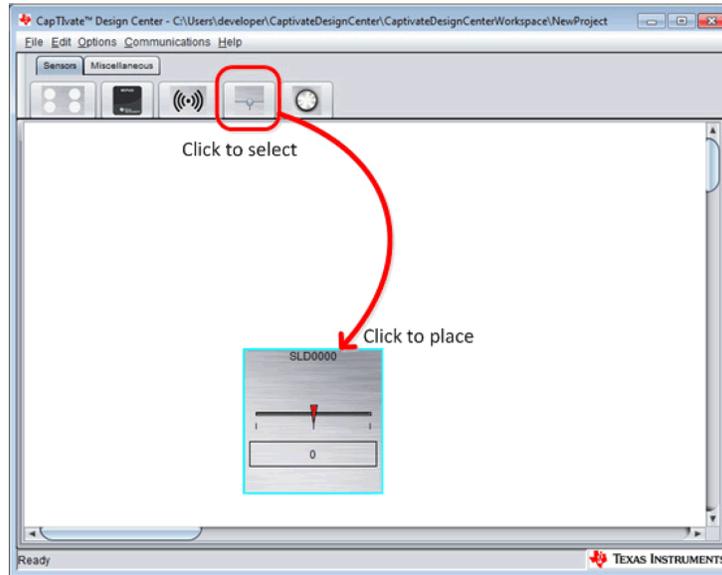
6.5 Rapid Development

Before developing your own hardware board, you can rapidly create a project, configure touch capabilities parameters, and start validation or simple development based on the MSP hardware validation platform and CapTivate Design Center.

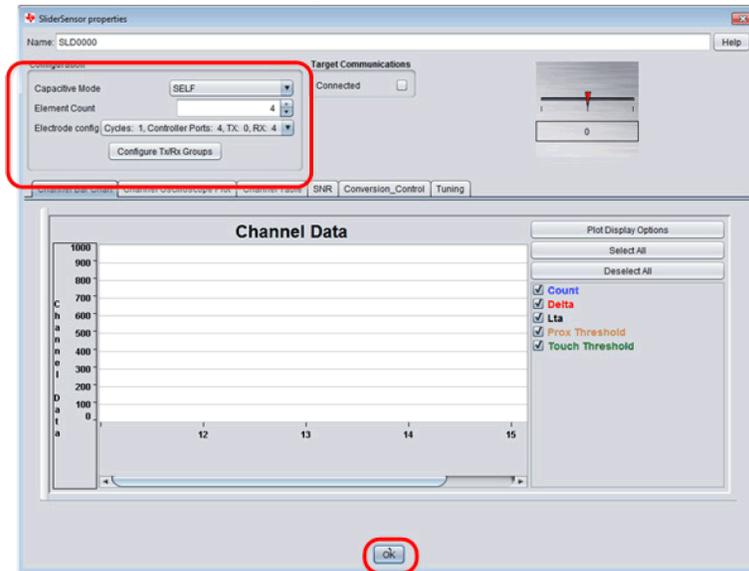
1. Create the new project on CapTivate Design Center.



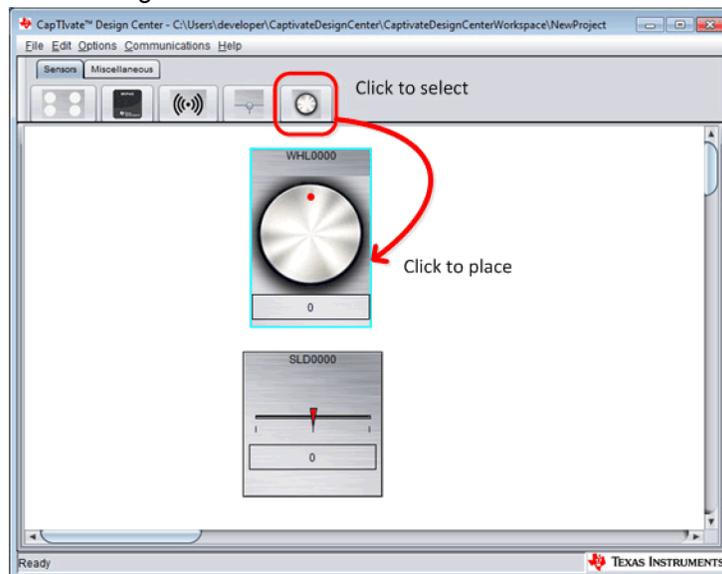
2. Place a slider sensor in the design area.



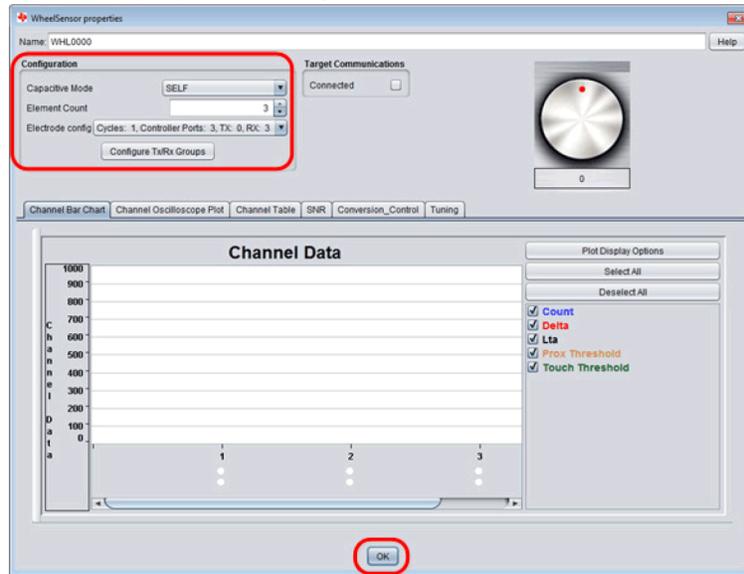
3. Configure the slider sensor on properties dialog.



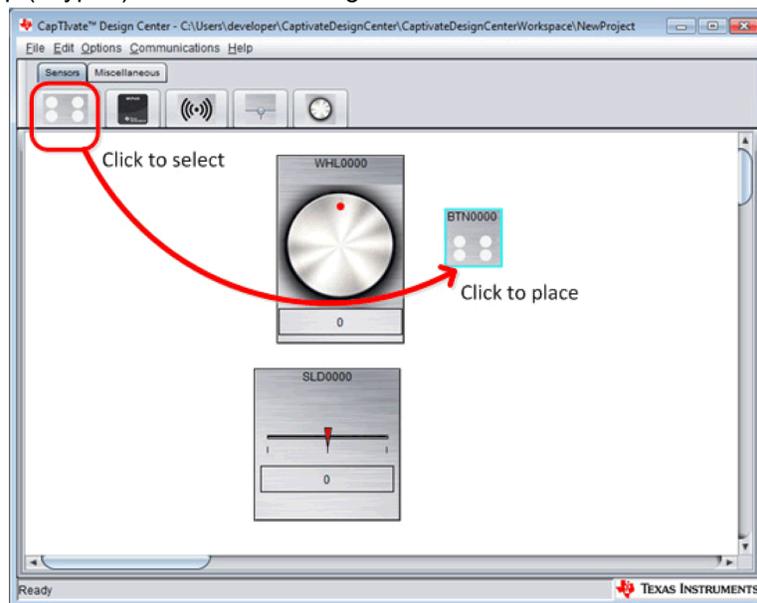
4. Place a wheel sensor in the design area.



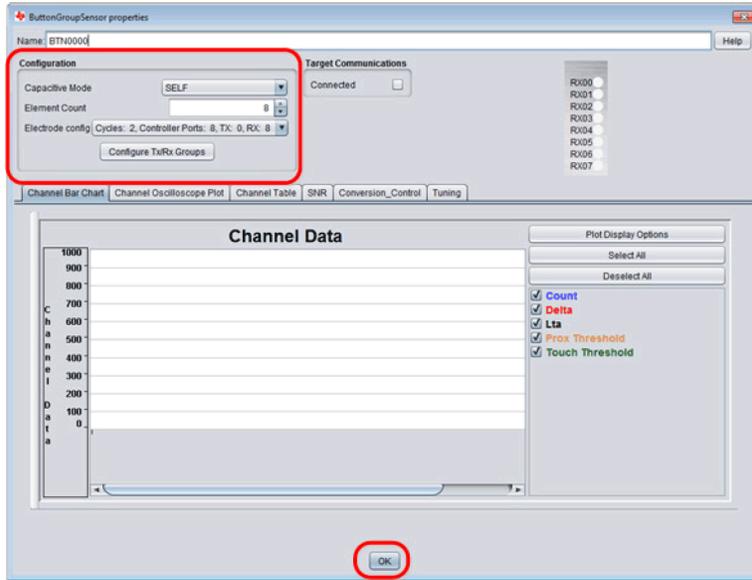
- Configure the wheel sensor on properties dialog.



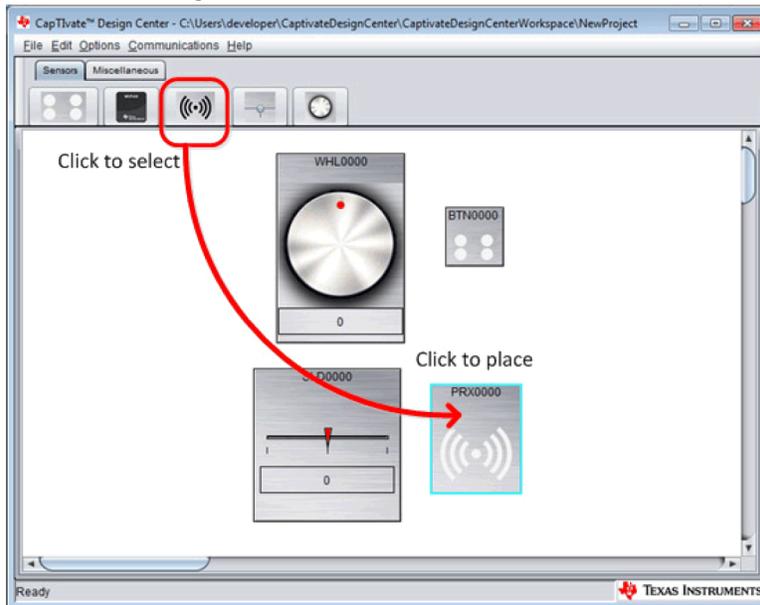
- Place a button group (keypad) sensor in the design area.



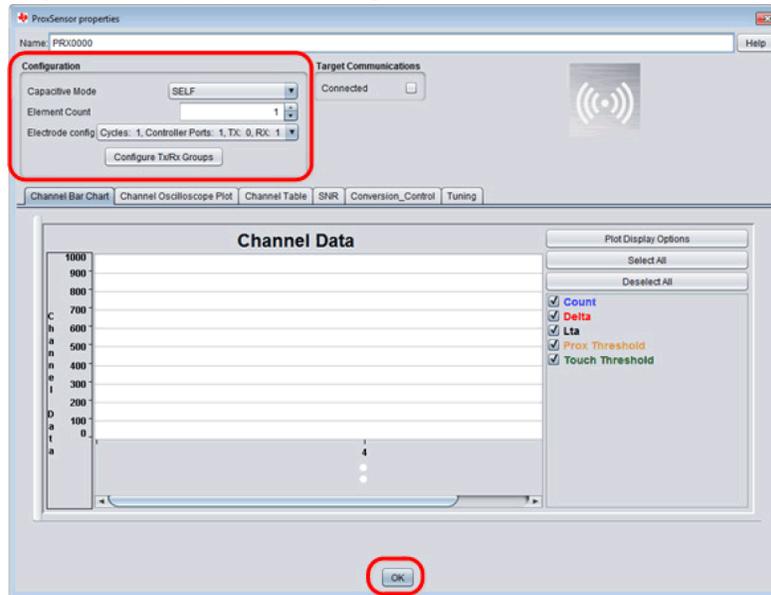
7. Configure the button group sensor on properties dialog.



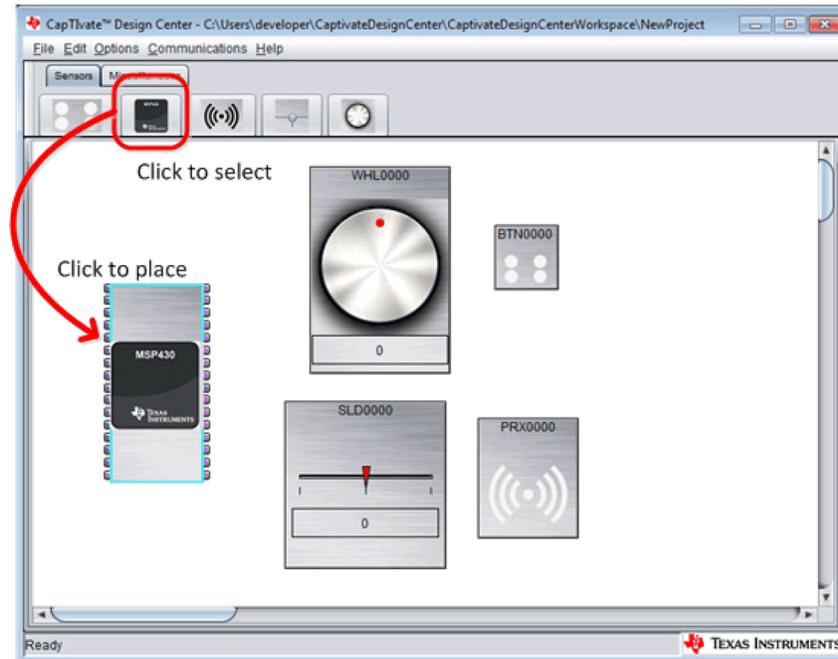
8. Place a proximity sensor in the design area.



9. Configure the proximity sensor on properties dialog.

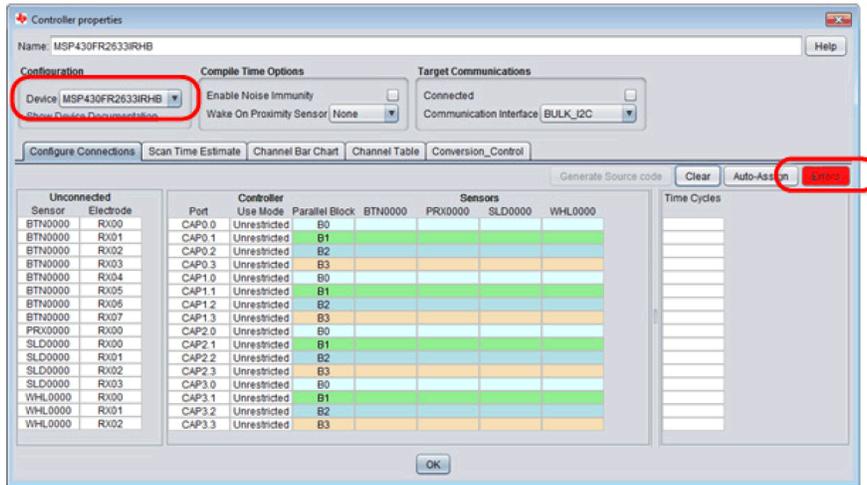


10. Select and place the MSP430 controller.

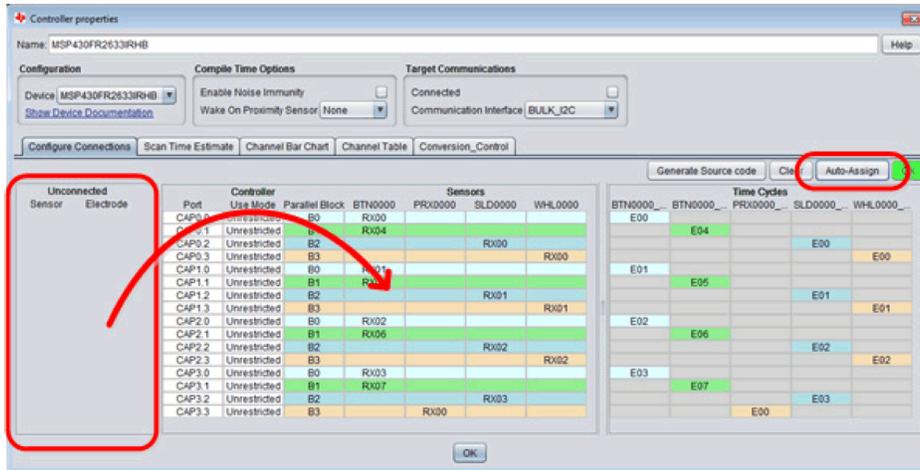


11. Connect sensors to MSP430 capacitive touch I/O ports.

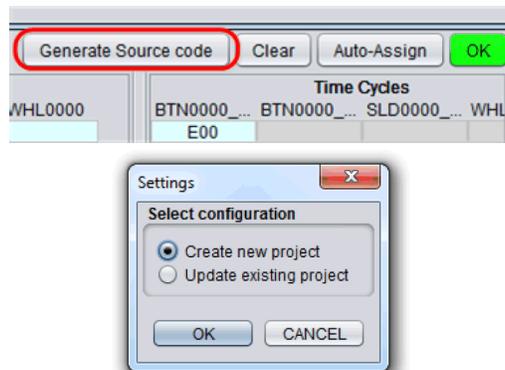
Double-click on the MSP430 controller object in the design area to display its properties. Configure the MSP430 controller as MSP430FR2633IRHB (32-pin QFN package). Note that the “Errors” LED is red, indicating that there are still unconnected sensor ports.



Select the “Auto-Assign” button to automatically assign all the sensor ports to appropriate ports on the MSP430. Note that the “Errors” LED turns green and “OK”, indicating that all sensor ports have been assigned to controller ports.



12. Generate source code.



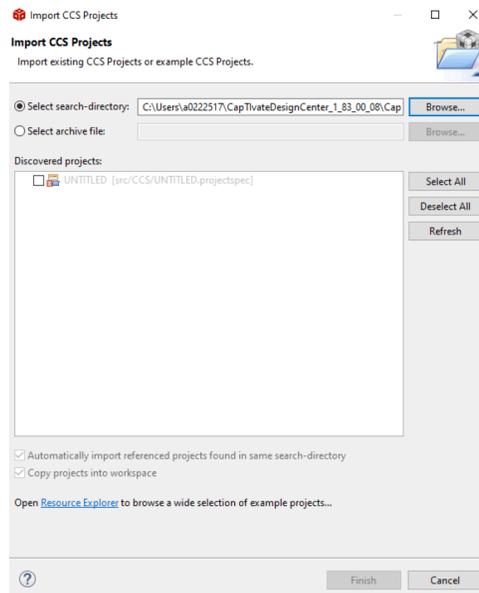
Saving the source code on default location of CapTIvate Design Center is recommended.



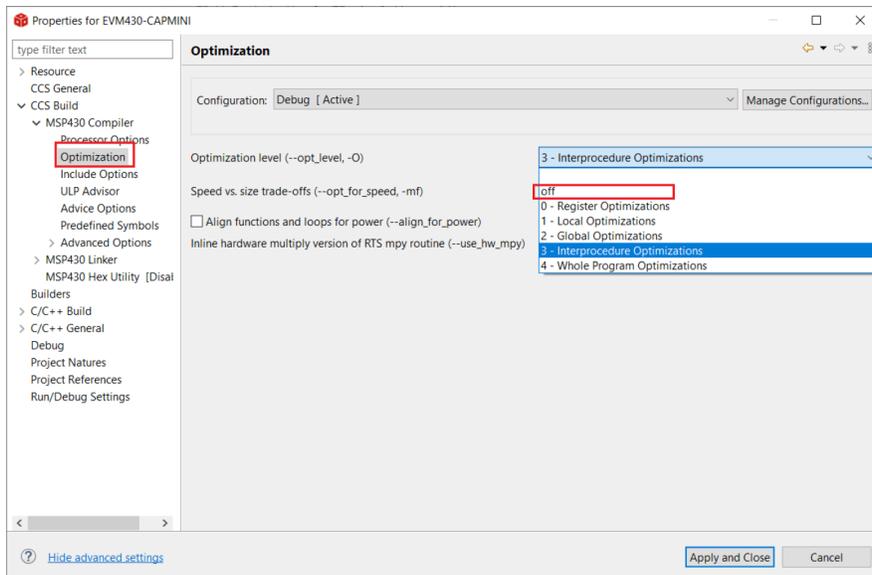
13. Import the project to CCS and start development

After completing the rapid evaluation process according to [Section 6.4](#), the project generated after adjusting the parameters on CapTlvate Design Center is imported into CCS to add system functionality.

Note: If the CCS icon is grayed out, there is a project with the same name in the workspace or CCS directory.



To reduce code size, select the highest level of code compilation optimization. TI recommends turning off optimization during development.



7 References

1. [MSP430 MCUs Development Guide Book](#)
2. [CapTIvate™ Technology Guide](#)
3. [Capacitive Touch Design Flow for MSP430™ MCUs With CapTIvate™ Technology](#)
4. [Sensitivity, SNR, and Design Margin in Capacitive Touch Applications](#)

8 Revision History

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

Changes from Revision * (May 2021) to Revision A (July 2021)

Page

• Changed the values in the "Passed CNI test (EMC) level based on the button structure" row in Table 2-1 Comparison of Self-Capacitive and Mutual-Capacitive Detection	5
• Added Figure 2-6 and related description in Section 2.3 TI's Capacitive Touch Technology	6
• Removed former Equation 8 in Section 5.1.3 Relationships of Parameters in CapTIvate Technology	19
• Updated the first list item in Section 5.1.5 CapTIvate MCU Communication Modes	21
• Removed duplicate figure in Step 11 of Section 6.5 Rapid Development	42
• Added items to Section 7 References	49

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale (<https://www.ti.com/legal/termsofsale.html>) or other applicable terms available either on [ti.com](https://www.ti.com) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2021, Texas Instruments Incorporated