CapTIvate™ Touch Getting Started Manual

Revision History

Version	Date	Author	Description
1.0	4/6/2020	Eason Zhou/Xiaodong Li	First version

Terms and Abbreviations

Abbreviation /Term	Meaning / Explanation
FRAM	Ferroelectric RAM (FeRAM, F-RAM or FRAM)
GUI	Graphical user interface
IDE	Integrated development environment
BSL	Bootloader
JTAG	JTAG(named after the Joint Test ActI/On Group) is an industry standard for verifying designs, testing printed circuit boards programming and debugging after manufacture
SBW	2-wire Spy-Bi-Wire interface, a typical JTAG interface for MSP430
MSP	Mixed Signal Processor
NVM	Nonvolatile memory

Reference Manual

CapTIvate™ Technology Guide

Capacitive Touch Design Flow for MSP430™ MCUs With CapTIvate™ Technology

Directory

1 Introduction

1.1 Overview

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 to perform:

1. Wear and reliability decrease after prolonged use.
2. There is a gap between the front panel and the buttons, which is easy to be penetrated by moisture and cause defects.
3. Need to exert force to trigger.
4. Opening the front panel will increase the cost to a certain extent.
5. The button shape is relatively fixed.

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

This document is written according to the actual capacitive touch development process, also to help users quickly understand the full picture of TI CapTIvate™ capacitive touch technology. For the preliminary function evaluation, it is recommended to read Chapter 1, Chapter 2 and the appendix. For hardware and mechanical structure development, it is recommended to read Chapter 1, Chapter 2, and Chapter 3. For software development, it is recommended to read Chapter 2, Chapter 4 and Appendix.

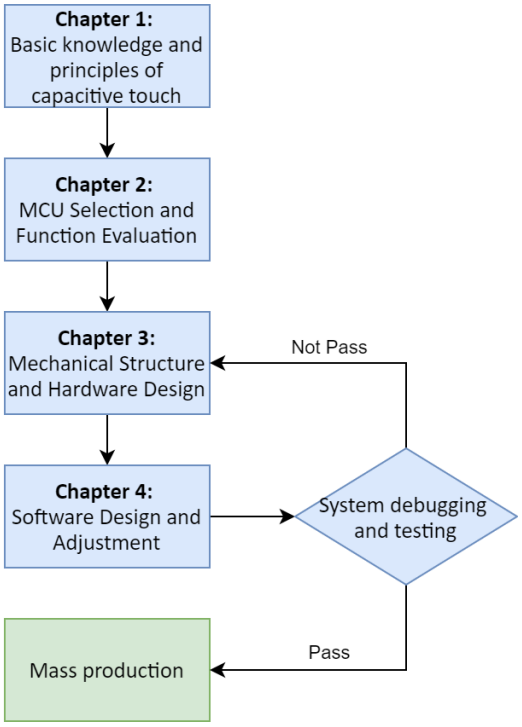


Figure 1-1 Capacitive touch development process

1.2 Introduction to related vocabulary

CapTIvate™: TI's capacitive touch design system.

Base capacitance: the parasitic capacitance of the sensor before the finger touches it.

CAP I/O: The pin on MSP430 dedicated to realize capacitive touch function.

LPM0/3/4: Different low-power modes of MSP430. For specific power consumption, please refer to the datasheet of the relevant device.
Rx: In mutual-capacitance mode or self-capacitance mode, the pin/electrode responsible for charging from the parasitic capacitance to the internal reference capacitance of the MCU.
Tx: In mutual-capacitance or self-capacitance mode, the pin/electrode responsible for charging the parasitic capacitance.

2 Basic knowledge and principles of capacitive touch

A common capacitive touch sensor is shown in Figure 2-1, and generally uses copper on the PCB as electrodes. Structurally, the top layer will be covered with a non-conductive protective layer, such as glass or plastic, and glued to the PCB. In addition, there will be a grid ground around the sensor.

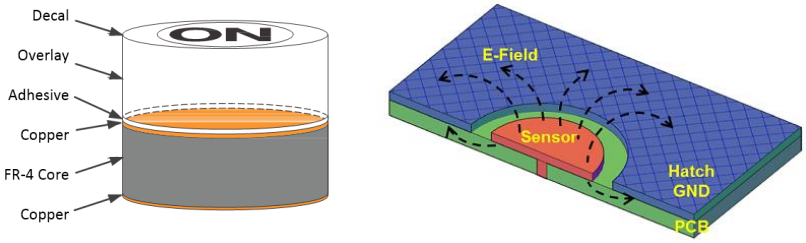


Figure 2-1 Capacitive touch structure

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

2.1 Self-capacitive detection

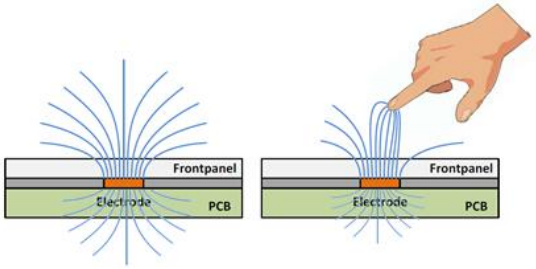


Figure 2-2 Self-capacitive detection

Taking the simplest single button as an example, the detection schematic diagram of the self-inductive capacitor is shown in Figure 2-2, and the detection model is shown in Figure 2-3. 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 will be introduced into the entire loop, thereby **increasing** the capacitance of the button to ground.

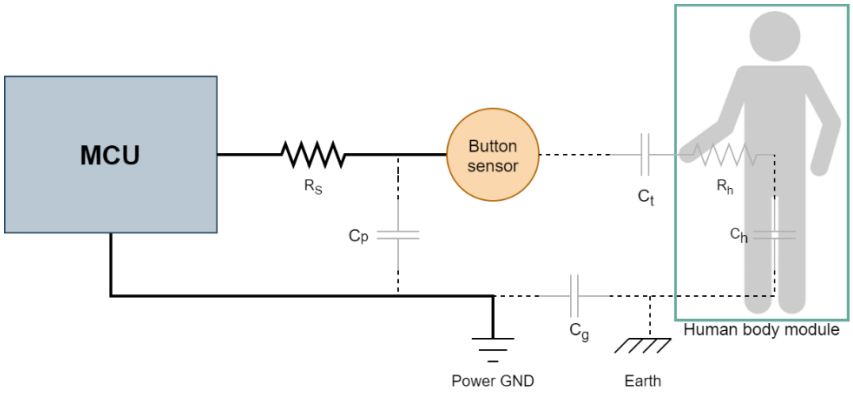


Figure 2-3 Self-capacitive detection model

Note: The solid line indicates the actual routing, and the dashed line indicates the non-real 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. The value is about 100pF~200pF.
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. The equivalent capacitance of the button to the power line ground is shown in Equation 1-1. Sensitivity can be characterized as the ratio between the capacitance change caused by the touch and the base capacitance, as shown in Equation 1-1. Among them, C_h is larger than C_g and C_t, so it can be ignored.

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

$$Sensitivity = \frac{C_{equal} - C_{base}}{C_{base}} = \frac{C_t||C_h||C_g}{C_p} \approx \frac{C_t||C_g}{C_p} \quad (1\text{-}2)$$

The calculation formula of parallel plate capacitance is:

$$C = \varepsilon_r \varepsilon_0 \frac{A}{d} \quad (1\text{-}3)$$

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 formulas 1-2 and 1-3 that the methods to improve the sensitivity are: 1) Reduce the thickness of the cover plate to increase the ε_r and C_t; 2) Reduce the density of the grid ground, or increase the PCB's Thickness to reduce C_p; 3) Since C_t is of the same order of magnitude as C_g, connect the power ground to the earth ground reasonably to increase C_g; 4) Increase the contact area A between the finger and the sensor pad covering layer to increase C_t. It should be noted that you cannot increase the electrode indefinitely. The main reason is that the maximum effective area of the parallel plate capacitance Ct is the same as the finger touch area. In addition, it will also increase C_p, resulting in a decrease in sensitivity.

2.2 Mutual-capacitive detection

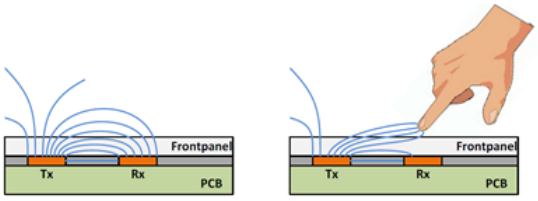


Figure 2-4 Mutual-capacitive detection

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. The biggest feature of mutual-capacitive detection is that the influence of C_p between sensor and the power ground can be ignored. Take the simplest single button as an example, the detection model of mutual-capacitive capacitor is shown in Figure 2-5. When touched by a human hand, the C_{RT} becomes two $2C_{RT}$s, and C_{RTt}, Ct, C_h and C_g are introduced at the same time. Finally, the capacitance between the two electrodes is **reduced**.

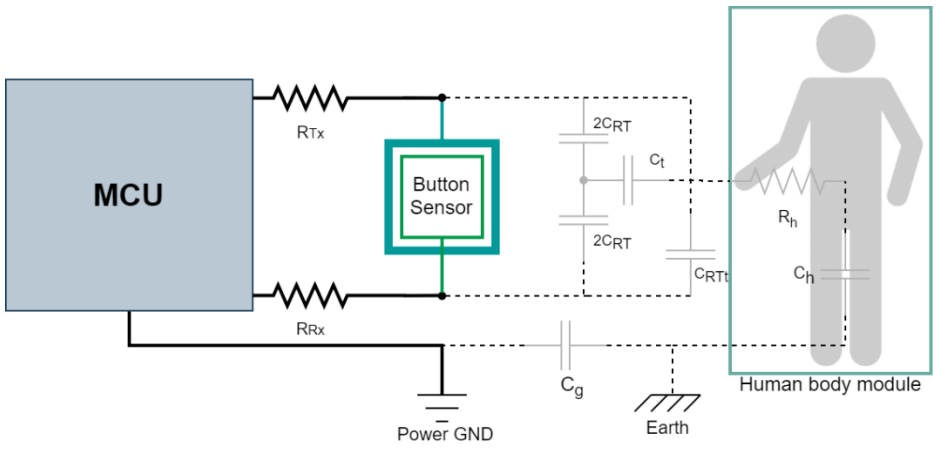


Figure 2-5 Mutual-capacitive detection model

Note: The solid line indicates the actual routing, and the dashed line indicates the non-real 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 2 C_{RT}, when touched by a human hand.

The equivalent capacitance between Tx and Rx is shown in Equation 1-4. Sensitivity can be characterized as the ratio between the capacitance change caused by the touch and the base capacitance, as shown in Equation 1-5.

$$C_{equal} = \frac{4{C_{RT}}^2}{4C_{RT} + C_t||C_h||C_g} + C_{RTt} \approx \frac{4{C_{RT}}^2}{4C_{RT} + C_t||C_g} + C_{RTt} \quad (1\text{-}4)$$

$$Sensitivity = \frac{C_{base} - C_{equal}}{C_{base}} \approx \frac{\frac{C_{RT} * C_t||C_g}{4C_{RT} + C_t||C_g} - C_{RTt}}{C_{RT}} \quad (1\text{-}5)$$

For mutual-capacitive touch, the main ways to improve sensitivity are: 1) Reduce the thickness of the cladding layer; 2) Increase the spacing between Tx and Rx to reduce C_{RT}. But pay attention that the sensitivity will decrease instead, when 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 about 1pF. But the base capacitance of self-inductance (capacitance value before touch) generally reaches several hundred pF. While for mutual-capacitive, it is about tens of pF. 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. The comparison between the two schemes is shown in Table 2-1.

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	10Vrms	3Vrms
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
Proximity sensing distance	>10cm	<3-4cm
Support touch screen function	No	Yes

2.3 TI's capacitive touch technology

TI's capacitive touch sensing technology CapTIvate™ is based on charge transfer collection. The principle includes: 1) Charging the sensor input capacitor C_{equal}; 2) Transferring the accumulated charge to the internal sampling capacitor C_{sample}. This process will continue to repeat 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 number will change. The MCU senses the occurrence of a touch event by comparing the numbers of different charge transfer cycles. MSP430 uses a current mirror to control the proportional relationship between the input current of C_{sample} and the discharge current of C_{equal}, so as to equivalently amplify C_{sample} and have a larger range.

3 MCU selection and function evaluation

3.1 Determine sensor requirements

Before start the capacitive touch project using MSP430, please understand the selected **sensor type**, **the number of sensors**, and **the occupied CAP I/O port**. Finally, select the appropriate MSP430 according to the corresponding requirements.

At present, capacitive touch sensors are mainly divided into the following types: button/proximity sensor, slider/wheel, and touch screen.

3.1.1 Button/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:

From Figure 3-1, 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 by a ring gap. Each button needs to occupy **1** MCU CAP I/O pin.

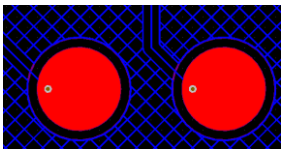


Figure 3-1 Self-capacitive button PCB diagram

Mutual-capacitive button:

In applications that require a large number of buttons, capacitive sensors can be arranged in a matrix, each button will be connected to two CAP I/Os, and one CAP I/O will be connected to a 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 row * Y column buttons need to occupy **X+Y** MCU CAP I/O pins.

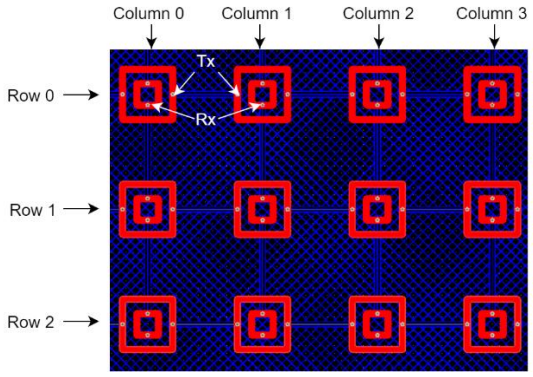


Figure 3-2 Mutual-capacitive button matrix PCB diagram

Self-capacitive proximity sensor:

Usually, the proximity sensor chooses the self-capacitive scheme. The main reason is that it has a longer detection distance than mutual-capacitive scheme, more flexible wiring and structure, and 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 selected. A self-inductive proximity sensor needs to occupy **1** CAP I/O pin.

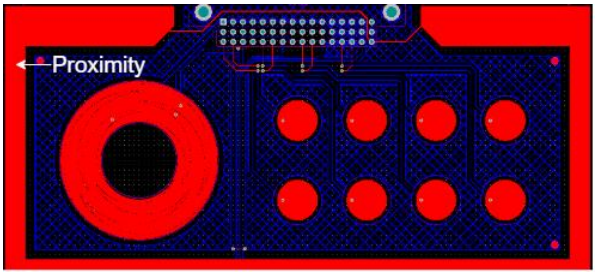


Figure 3-3 Mutual-induction proximity-sensing PCB diagram

3.1.2 Slider/wheel

Both the slider and the roller are one-dimensional linear sensors, and their implementation principles and structures are similar. Since the base capacitance of the mutual-capacitive scheme is low and the signal fluctuates greatly than self-capacitive scheme, it is less used in this structure. Therefore, only the self-inductance slider/roller is discussed, as shown in Figure 3-4. The basic structure of the slider/scroller is the buttons. By interlacing the buttons, when the human hand moves, because the contact area of the two adjacent buttons is different, the gradual signal can be detected. In addition, the two copper coatings at the beginning and the end of the slider should be connected together by wires and belong to the same button.

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

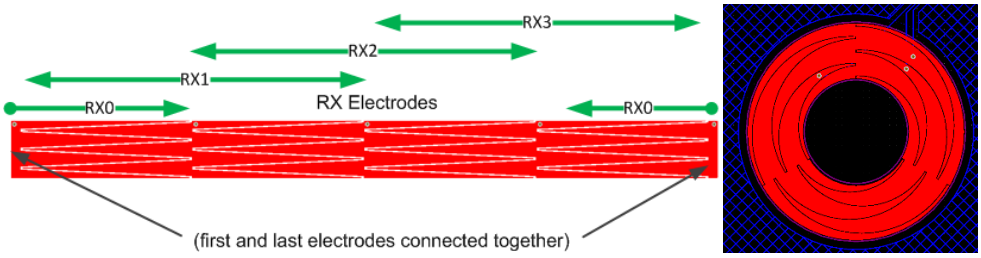


Figure 3-4 Self-capacitive slider/roller schematic diagram and PCB diagram

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, the gradual change signal can be detected on two adjacent buttons when the human hand moves. In the following figure 3-5, the intersection of 4 RX and 4 TX forms 16 measurement nodes. A single touch panel often occupies more than **8** CAP I/Os.

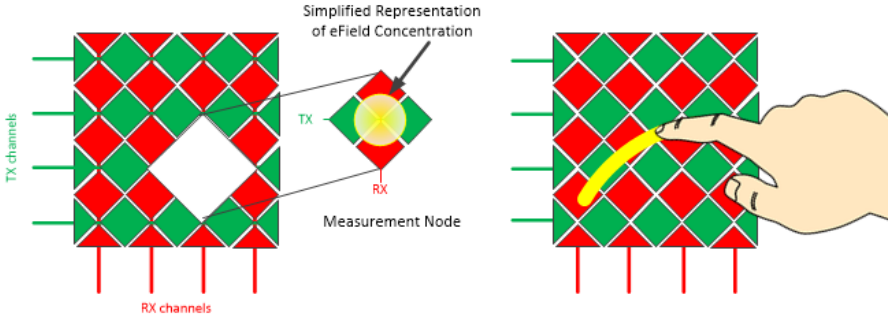


Figure 3-5 Schematic diagram of touch panel

3.2 MCU selection

Table 3-1 summarizes all MSP430s that support capacitive touch. According to the required CAP I/O and corresponding peripherals, the appropriate MSP430 can be selected. It should be noted that the capacitive touch code needs to occupy **3-6k** bytes of space according to different configurations, and the maximum output current of the I/O port under 3V power supply is only 5mA.

Table 3-1 MSP430 CapTIvate™ series device selection

Part number	Generation	CAP I/O	FRAM (kB)	RAM (KB)	ADC	Communication	Package Group
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

What needs to be added is that the CapTIvate™ series of MSP430 is mainly divided into two series. The first-generation product Gen1 focuses on cost performance, while the second-generation product Gen2 has more abundant peripherals and stronger CapTIvate™ module performance. The specific differences are shown in Table 3-2.

Table 3-2 Comparison of Gen1 and Gen2

Features	Gen1 device	Gen2 device	Advantages
Electrode charging voltage	VREG power supply: 1.5V	VREG power supply: 1.5V DVCC power supply: 2.7-3.6V	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-25°C)	19:1–36:1	28:1–42:1	Higher signal-to-noise ratio and lower temperature drift.

3.3 EVM development board selection and evaluation

In the previous evaluation of capacitive touch function, the software and hardware involved are shown in Table 3-3. To set up the entire development chain, GUI, IDE, SDK, Programmer, MCU, and sensor board are required. Part of the MCU board and sensor board are integrated.

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™-BSWP	Self-mode button/Wheel/slider/Proximity
				CAPTIVATE™-FR2633	CAPTIVATE™-PHONE	Mutual-mode button/Wheel/slider/Proximity
				BOOSTXL-CAPBUTTONPAD(FR2522)		12 mutual mode buttons and 1 proximity
				EVM430-CAPMINI (FR2512)		4 self-mode buttons

The software part of the development chain is:

- 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 part of the development chain is:

- Programmer: Support JTAG and USB HID to UART/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/FR265x series.
    - BOOSTXL-CAPBUTTONPAD is only used to evaluate specific 12 button panel solutions, or to evaluate MSP430FR2522.
    - EVM430-CAPMINI is only used to evaluate specific 4 self-inductive button schemes, or to evaluate MSP430FR2512.
- Sensor board: An evaluation board containing various sensors.
    - CAPTIVATE™-BSWP is used to evaluate TI's self-inductive solutions
    - CAPTIVATE™-PHONE is used to evaluate TI's mutual-capacitive solutions

For other CapTIvate™ related materials, including instruction documents, teaching videos, development boards, and application manuals, please refer to the section 5.2.1.1 of the MSP430™ MCUs Development Guide Book.
TI has configured corresponding GUI and CCS projects for all EVM boards. The default installation directory is: C:\Users\UserName\CapTIvate™DesignCenter_x_xx_xx_xx \CapTIvate™DesignCenterWorkspace\TI_Examples.
Users can refer to the steps in the appendix 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, it is recommended to read the first chapter to understand the basic principles of capacitive touch in order to better understand this chapter. 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 and compromise 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 formulas 1-3 that the dielectric constant and thickness of the covering material will affect the sensitivity of the capacitive touch. The dielectric constants of different materials are shown in Table 4-1.

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 + Epoxy)	4.2
PMMA (polymethyl methacrylate)	2.6 - 4

For panels stacked from different materials, the touch capacitance can be calculated equivalently according to the following formula:

$$C = \varepsilon_r \varepsilon_0 \frac{A}{d} = A\varepsilon_0 \sum \frac{\varepsilon_{ri}}{d_i} \quad (4\text{-}1)$$

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.

Since the dielectric constant of air is only about 1, it can be seen from the formula 4-1 that the presence of the variable phase thickens the panel, so the air between the sensor and the cover layer should be eliminated as much as possible. In addition, in many cases, the sensitivity of the buttons is low, mainly because the panel is too thick. Because the conductor will interfere 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

According to actual application requirements, various materials can be used to construct capacitive sensors. The following will briefly introduce the common sensor structure.

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. It should be noted that the flexible PCB is thinner than rigid PCB, which means that the distance between the button and the ground plane is relatively short. Usually, the density of the grid ground plane should be appropriately reduced.

4.1.2.2 Conductive washer/spring type sensor

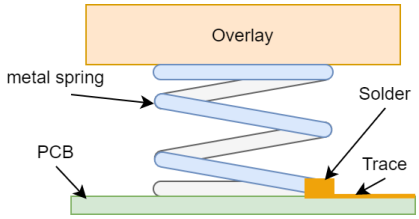


Figure 4-1 Schematic diagram of spring structure

This solution realizes the expansion of the touch button space through conductive material washers/springs. It is mainly suitable for situations where the touch front panel cannot be attached to the PCB, and it is also suitable for curved, inclined or irregular front panel solutions.

4.1.2.3 Electronic ink type sensor

The electronic ink-type sensor uses conductive ink to form a capacitive sensor. Generally, close integration with the front panel is achieved by spraying or printing. The main features are flexible patterns and shorter design cycles, which are suitable for irregular front panels. Another application scenario is the ITO liquid crystal display, which mainly utilizes the transparency characteristics of the ITO film. Since the film resistance of electronic printing ink is larger than that of copper, the series resistance needs to be appropriately reduced.

4.1.3 Mechanical design checklist

There are three main aspects of the mechanical structure that will affect the touch sensitivity: coating, housing, and surrounding devices. The unreasonable coating design mainly affects the capacitance change caused by finger touch. For the self-inductive solution, grounding the shell will improve the noise immunity of the product, but it will increase the capacitance to ground and reduce the sensitivity. For the 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	Avoid the use of conductive materials and conductive paints.
2		Thickness	10mm or thinner, 2-3mm is recommended, depending on the material and sensor size.
3		Cascade	Avoid gaps between the overlay and the buttons, use non-conductive adhesive materials.
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 away from the internal radiation noise source of the product as far as possible. Keep the PCB with button function away from sensitive devices as much as possible. 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 whether the MCU function is normal, and EMC anti-noise performance. Table 4-3 is a checklist for schematic design. For specific design solutions, please refer to the corresponding device datasheet, or refer to the EVM schematic diagram in Chapter 3.3 of this manual.

Table 4-3 Schematic Design Checklist

Number	Classification	Component	Recommendation	
1	Smallest system	Reset and programming circuit	Add 47kΩ pull-up resistor and 1nF pull-down capacitor to Reset pin.	
2		Power supply circuit	Add 10μF and 0.1μF capacitors to VCC and GND, and place them close to the MCU.	
3	Capacitive touch	VREG filter circuit	Add 1μF capacitance to ground close to VREG pin, ESR ≤ 200mΩ.	
4		Series resistance on CAP I/O	Add a 470Ω-10kΩ 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. **Note:** In addition to buttons and proximity sensors, scroll wheels, sliders, and touch panels have sequence requirements for scanning CAP I/O. It is strongly recommended to use the Auto-Assign function to assign scroll wheels, sliders, and touch panel CAPs after configuring the buttons. I/O pin.	
6	EMC anti-noise (Optional)	EMC filter capacitor	For mutual-capacitive applications, add 68pF of capacitance to ground between the series resistance on the RX pin and the sensor electrode.	
7		TVS diode	Add a 3.3V 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 pull-up resistor	Add 2.2kΩ pull-up 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/253x/267x: 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/253x/267x: P1.4: UCA0 UART TXD P1.5: UCA0 UART RXD
12		Boot Loader (BSL)	For BSL pin definition, please 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.	

For the EMC anti-noise design, during the preliminary evaluation, it is recommended to reserve the welding position of the corresponding component as needed, and select the welding according to the final test result. What is important to explain is the choice of series resistance on CAP I/O. Because the series resistance will affect the base capacitance and the capacitance change caused by the touch at the same time, the series resistance itself will not affect the sensitivity of the button. Since it forms a low-pass filter with parasitic capacitance, a larger series

resistance can increase the noise immunity of the button. But it should be noted that too large resistance will affect the charge transfer time. The charge and discharge waveforms on the CAP I/O are shown in Figure 4-2. This will directly lead to a decrease in the sensitivity of the button, which is especially obvious in a self-inductive solution with a larger base capacitance of the sensor. At this time, the charge transfer period can be extended and the "Frequency Divider" parameter in the GUI can be modified to solve the problem.

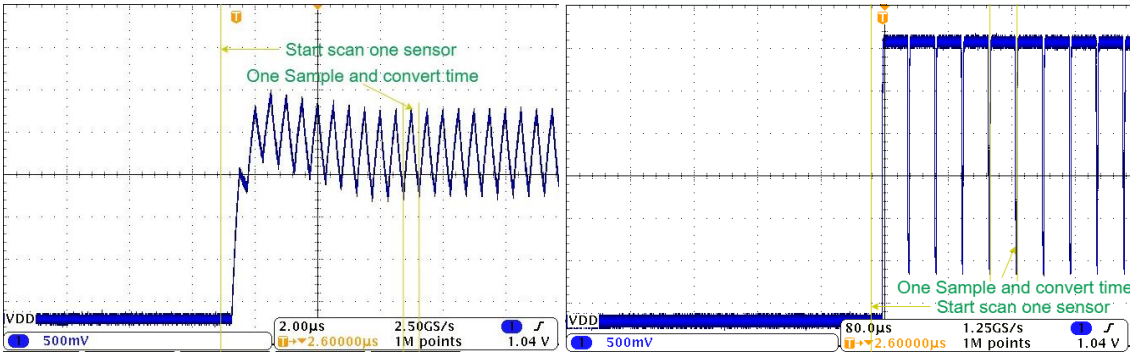


Figure 4-2 Incomplete and complete charge transfer cycle (GEN2)

4.2.2 PCB layout

The starting point of PCB layout and mechanical structure design is the same, one is to improve signal strength, and the other to reduce EMC problems. Table 4-4 is the PCB layout checklist. For specific examples, please refer to TI EVM board CAPTIVATE™-BSWP(Self-inductive) and CAPTIVATE™-PHONE(Mutual-capacitive type).

Table 4-4 PCB layout checklist

Number	Component		Recommendation
1	Button (self-capacitance)	Shape	Solid circle or square.
		size	The diameter/side length is 10mm -12mm.
		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 0.5mm, Tx width is 1mm, and the distance between Rx and Tx is 0.5mm.
		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/roller (Self-capacitance)	Shape	Please refer to Automating Capacitive Touch Sensor PCB Design Using OpenSCAD Scripts
		Size	The electrode width is 10-12mm, 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.
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 10mil-20mil spacing with grounding copper. No sharp angles. Reduce via holes, via hole diameter 10mil. 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	se grid-like copper plating (line width 8mil, grid width 64mil, angle 45°). Self-capacitive design, double-sided copper-clad, selective copper-clad directly under the button. Mutual capacitance type design, can choose only the bottom surface of copper.
8	Series resistance, TVS tube, EMC filter capacitor placement	Place within 10 mm of MCU pin.
8	Moisture and liquid resistance	Self-inductive solution: It is recommended to reduce the grounding copper and improve the touch sensitivity to reduce the parasitic capacitance change caused by water. Mutual-capacitive scheme: reference Liquid Tolerant Capacitive Touch Buttonpad Reference Design

In terms of sensitivity, buttons and slider wheels with a width similar to a finger touch, and narrower and shorter sensor traces can enhance the capacitance change caused by touch. Fewer vias and thinner copper can reduce the size of the capacitance to ground. However, grounding copper also directly affects EMC's anti-noise ability. The specific laying range and grid density need to be adjusted according to actual needs. Since capacitive touch is both a source of interference and a sensitive device, for EMC considerations, in addition to adding protection devices, it is recommended to keep away from other signal lines or devices during layout.

5 Software design and parameter tuning

The software development of capacitive touch is more complicated and requires software tools such as GUI and IDE. The development process will include setting parameters and function development. Therefore, before entering this chapter, it is recommended to read chapter 3.3 and appendix of this manual to understand the entire capacitive touch development chain, and to be familiar with the basic operations of GUI and IDE. For the relevant technical information of the MSP430 device itself, please refer to MSP430™ MCUs Development Guide Book.
The actual 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. This chapter will be expanded in sequence with these 5 phases.

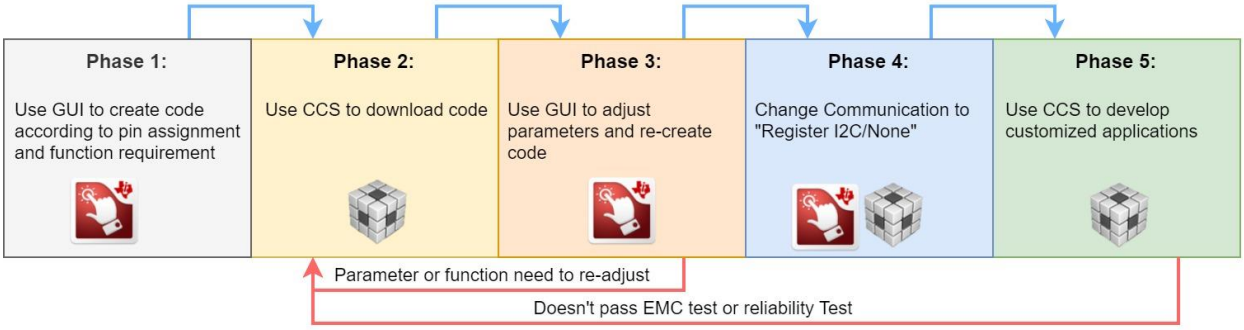


Figure 5-1 Software design process

5.1 Concepts required for CapTIvate™ software development

Before entering the various stages of development, I will first introduce several important concepts of CapTIvate™ to help users understand CapTIvate™s working methods, working principles, and the correlation between parameters. For more information, please refer to Technology chapter in user guide.

5.1.1 CapTIvate module and GUI function description

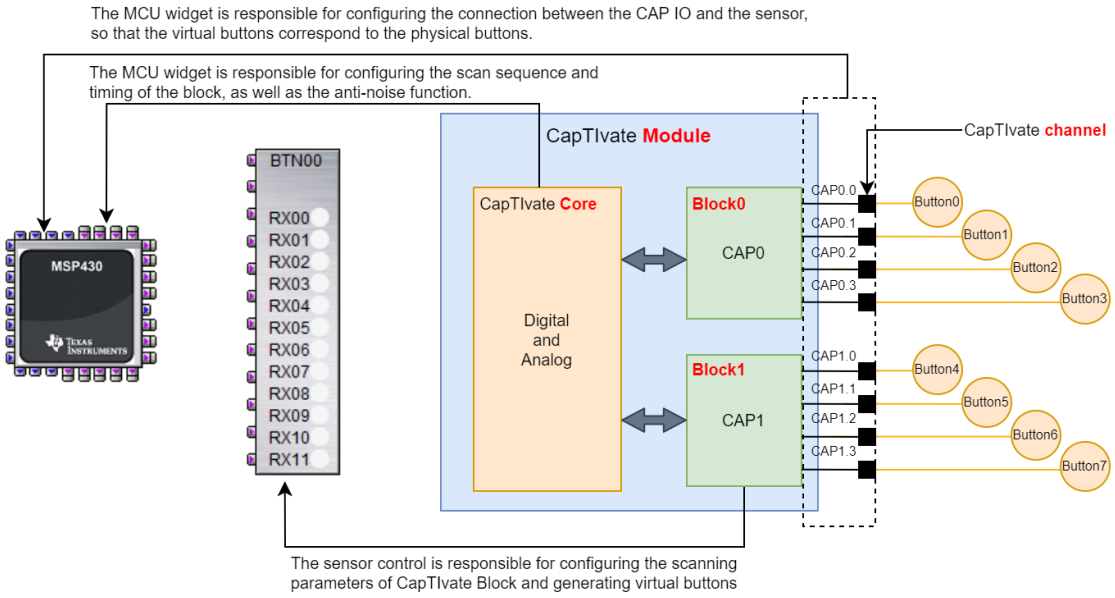


Figure 5-2 Correspondence diagram between GUI and CapTIvate module

The CapTIvate module consists of a Block and a CapTIvate Core. Block is responsible for scanning the external capacitance. CapTIvate Core is responsible for the scanning control of the block and the anti-noise function. It

should be noted that physically, one Block corresponds to four CAP IOs. This also means that only one CAP IO can be scanned under one block at the same time, so a reasonable configuration of the button scanning sequence can save a certain amount of time.

The most important part of GUI is MCU widget and sensor widget. These two controls are responsible for the configuration of CapTIvate Core. The corresponding relationship between GUI and CapTIvate module is shown in Figure 5-2. The sensor widget is mainly responsible for the control of the Block. MCU widget is mainly responsible for the control of CapTIvate Core.

5.1.2 MCU working mode

CapTIvate™ MCU has a total of two working modes defined, Active mode and Wake-on-Prox mode. For the power consumption in different modes, please refer to the datasheet of the corresponding device. This part of the function is configured in the MCU widget. As shown in Figure 5-3, in Active mode, the CPU will control Cap Peripheral to perform sensor scanning, and then the CPU will enter the LPM mode after sleeping. After waiting for "Acitve scan rate ms", the Timer in Cap Peripheral will wake up the CPU to scan the sensor again, and so on.

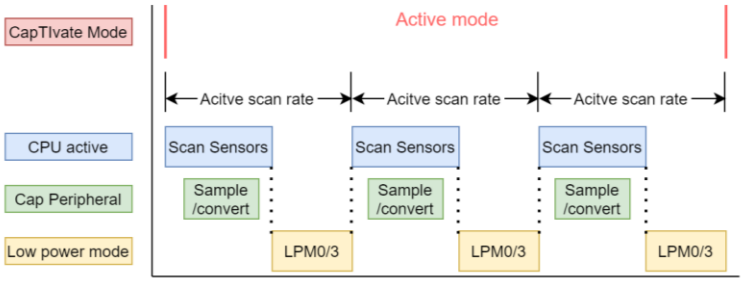


Figure 5-3 MCU status in Active mode

The Wake-on-Prox mode mainly uses the Cap peripheral's ability to work out of the CPU, and achieves lower power consumption than the Active mode by turning off the CPU for a long time. Since no CPU is involved, it means that 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 will work in Active mode, and multiple buttons will be scanned in turn. If no touch occurs, wait for the "Inactivity Timeout" sensor scan period, then enter Wake-on-Prox mode from Active mode. Turn off the CPU, then approach the wake-up sensor to scan at the "Wake On Prox Mode Scan Rate" scan interval. At this time, MCU can choose to enter LPM4 to further reduce power consumption.

After waiting for the "Wakeup Interval" scan interval (usually a few minutes), the MCU wakes up and enters the Active mode, and updates the parameters of all buttons to deal with environmental drift. Then enter the "Inactivity Timeout" count. After the counting is finished, MCU will enter the Wake-on-Prox mode again. So reciprocating cycle.

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

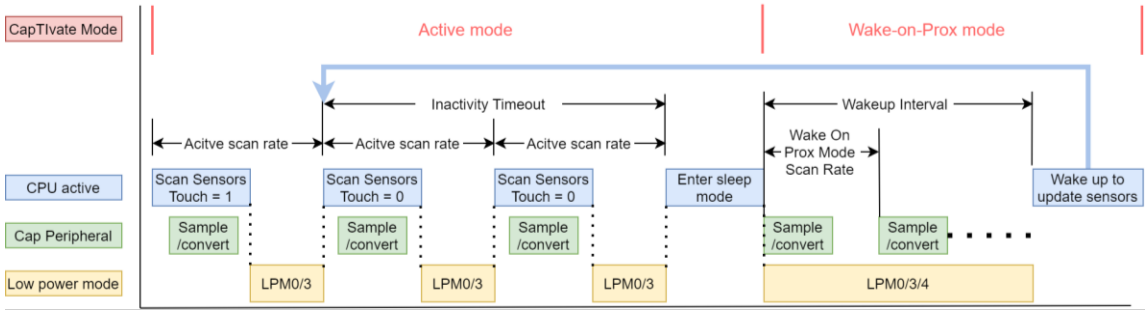


Figure 5-4 MCU status in Wake-on-Prox mode

5.1.3 The relationship of important parameters in CapTIvate™

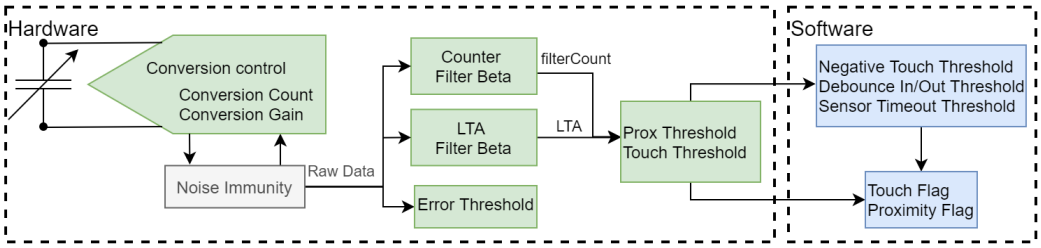


Figure 5-5 Schematic diagram of parameter relationships in CapTIvate™

The parameter structure in the entire CapTIvate™ is shown in Figure 5-5. The whole is 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.

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. It should be noted that for GEN1, part of the anti-noise function will be implemented by software.

The output data will first be judged by Error Threshold. Then filterCount (default intensity 1) is generated through IIR filtering of different intensities, which is used to characterize real-time capacitance changes, and LTA (Long time average, default intensity 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 human hand touch. The relationship between the capacitance change caused by the touch and the filterCount and LTA is shown in Equation 5-3. For self-capacitance detection, Delta is a positive value, and for mutual-capacitance detection, Delta is a negative value.

$$Delta = \text{filterCount} - \text{LTA} \tag{5-1}$$

$$C = \alpha * \frac{\text{Gain}}{\text{filterCount}} \tag{5-2}$$

$$\Delta C_{touch} = C_{touch} - C_{base} = \alpha * \text{Gain}\left|\frac{1}{\text{LTA} + \text{Delta}} - \frac{1}{\text{LTA}}\right| \tag{5-3}$$

The entire signal change process is: when power is on, the calibration function will calibrate 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 will change rapidly, and the Delta will increase from 0. When the Prox Threshold is triggered, the LTA Filter will be closed and the LTA value will remain 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 will cause the LTA to change significantly, and the LTA will drift 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 will be 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 formulas 5-4 and 5-5. In addition, it can be seen from formula 5-2 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.

$$|Delta| = \text{Prox Threshold} \tag{5-4}$$

$$|Delta| = \text{LTA} * \frac{\text{Touch Threshold}}{128} \tag{5-5}$$

Other Thresholds will determine 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 it. At this time, the touch signal of the human hand will be counted as the base capacitance of the environment. At this time, removing the finger will generate a negative signal. When the negative signal reaches the Negative Threshold, system calibration will be triggered. Debounce in/out threshold will introduce a signal anti-jitter mechanism to combat noise, which will delay 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 will recalibrate the environmental base capacitance.

5.1.4 Object structure in CapTIvate™

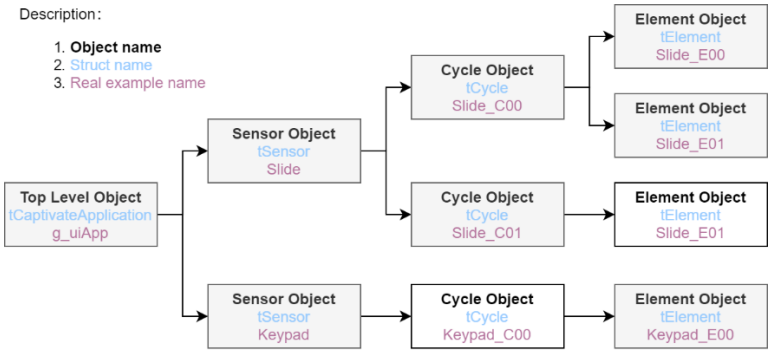


Figure 5-6 The object structure in CapTIvate™

There are four types of objects in CapTIvate™, 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 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 of this chapter.
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 will be multiple Sensor objects in a system.
Since one capacitance detection module of CapTIvate™ 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 mode

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

- None: Disable the communication module. Customers need to develop communication protocols to communicate with Host MCU by calling library functions. If you want to further save code space, you can choose register-level code in MSPWare as a reference for development.
- UART/BUCK_I2C: It is used to communicate with CapTIvate™ Design Center through UART/I2C to realize GUI adjustment of capacitive touch. Supports richer communication protocols, the bus will transmit a large amount of low-level information, which is generally used for online parameter adjustment. The UART communication module occupies about 1.5k of code, and the BUCK_I2C communication occupies about 2k of code.
- Register_I2C: It is used to communicate with Host MCU through I2C to realize Host MCU's adjustment of capacitive touch parameters. There are fewer debugging commands supported, occupying about 1.4k of code.

The whole use logic is that the user first selects the "UART/BUCK_I2C" communication mode to cooperate with 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.

Phase 1： GUI configuration

Please follow the instructions below to configure the relevant parameters. For more information, please refer to Design Center GUI chapter in user guide.

5.1.6 GUI main interface operation

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 will be involved in Phase2.

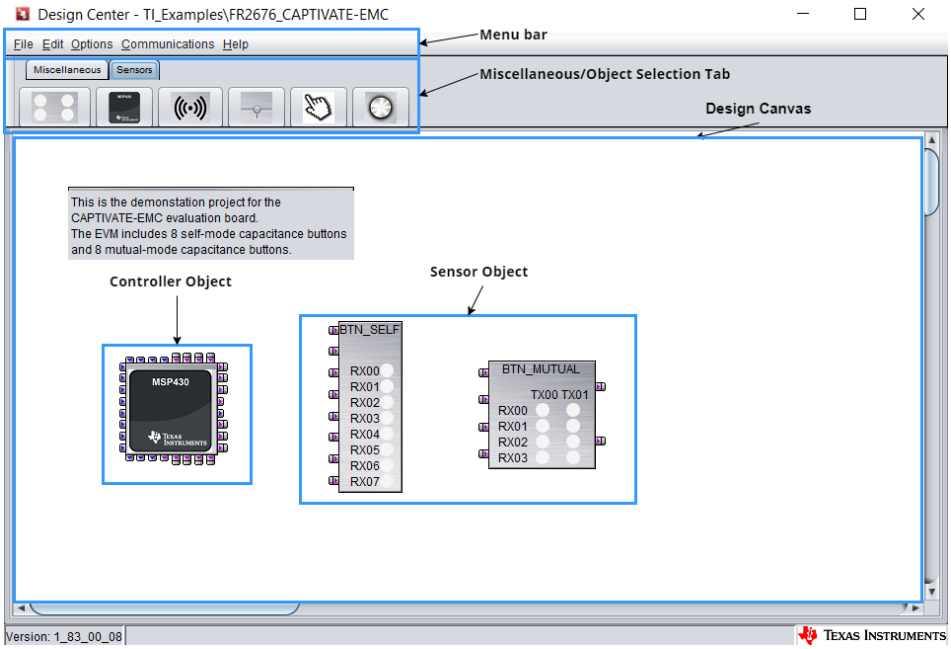


Figure 5-7 GUI main interface

Table 5-1 Parameters to be set in the GUI main interface

Box in picture	Operation directory	Operation content	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	

5.1.7 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 will be involved in Phase2.

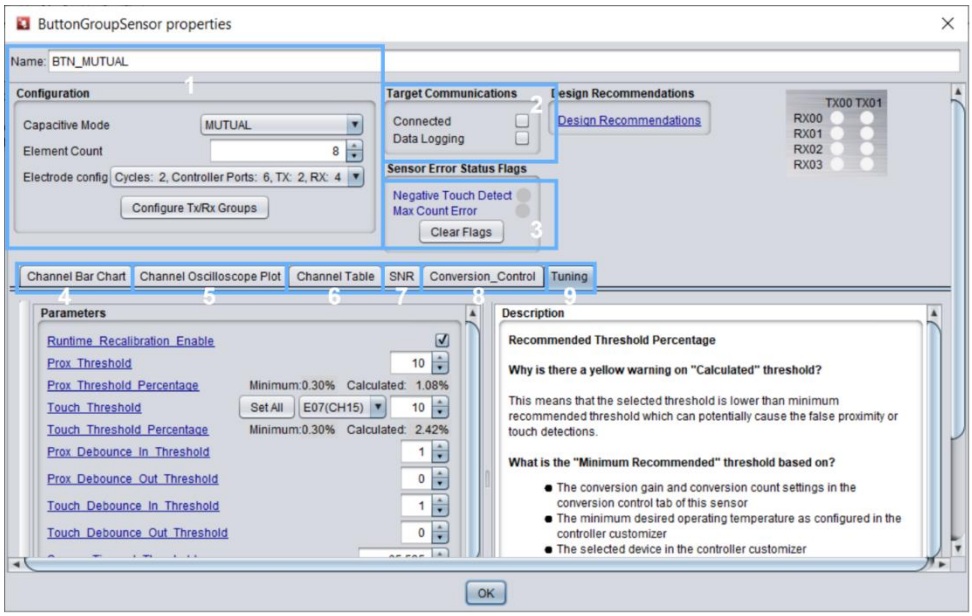


Figure 5-8 Sensor widget interface

Table 5-2 Parameters to be set in the sensor widget

Box in picture	Operation directory	Operation content	Recommended value	Comment
No. 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	

5.1.8 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 will be involved in Phase2.

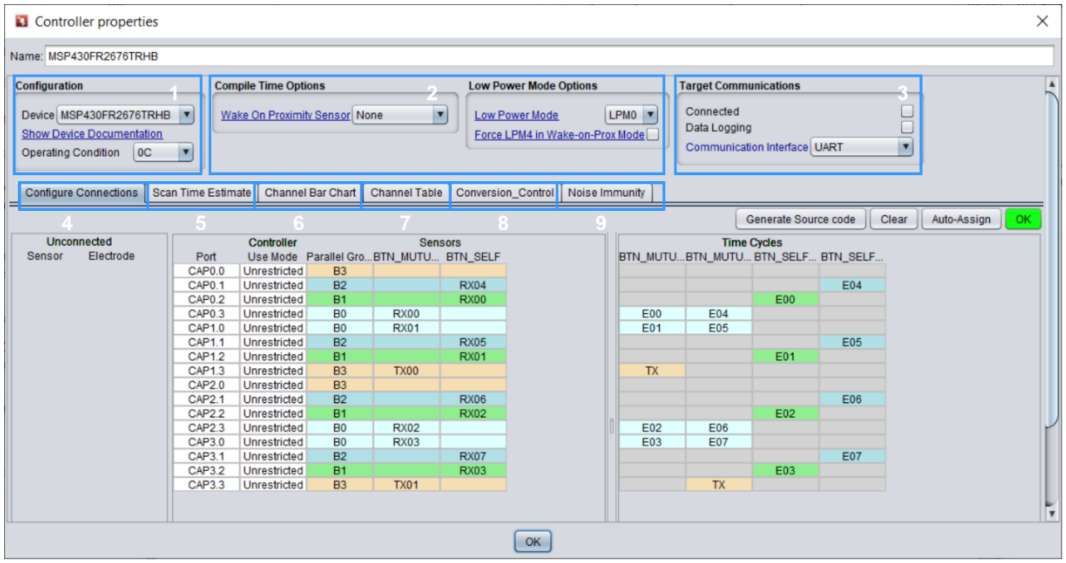


Figure 5-9 MCU widget interface

Table 5-3 Parameters to be set in the MCU widget

Box in picture	Operation directory	Operation content	Recommended value	Comment
No. 1	Device	Select the corresponding device and package.	None	
	Operating Condition	Choose the actual working temperature.	25C	Temperature will affect the noise of the device and affect the Threshold Percentage in the sensor widget.
No. 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 corresponding low-power mode according to your needs.	Can use default configuration	PM3 has lower power consumption than LPM0, but it cannot actively receive UART data at this time, 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.
No. 3	Communication Interface	Choose UART or BUCK_I2C to communicate with GUI according to actual design.	None	Please refer to section 4.2.1 of this manual for the communication pins and IRQ PIN used.
No. 4	Configure Connections	According to the actual button design, map the virtual Tx and Rx pins to the CAP IO.	Can click “Auto-Assign”	In addition to the button and proximity sensor function, it is recommended to select the Auto-Assign function.
No. 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".
No. 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.

No. 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 will be larger and the scanning time will increase.
	Self/Mutual mode Oversampling	Adjust oversampling level.	Can be unadjusted	Can improve the level of noise immunity, but oversampling will increase the scan time of a single cycle.
	Dynamic Threshold Adjustment	Enable dynamic adjustment of proximity sensing and touch threshold.	Enable	After enabling, it can be guaranteed that the sensitivity will change with the noise level, but when encountering sudden noise, the threshold change may lag behind the noise change.

It should be noted that the "Configure Tx/Rx Groups" in the sensor widget can know the Tx and Rx corresponding to the virtual button, and the "Configure Connections" in the MCU widget can know 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. Since creating a new project will **delete** all files in the current folder, if you don't put it in the default directory, please create a new folder and then put it in the project.

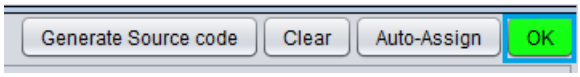


Figure 5-10 Schematic diagram of the OK button

5.2 Phase 2： Download code

The download tool supports CCS and IAR. Please refer to the Loading and running generated projects chapter in User Guide for specific methods of downloading projects. If there is a problem that cannot be imported, please check whether there is a project with the same name under the CCS workspace. For IAR, please make sure to download the latest version to support all capacitive touch related MSP430. If you download it for the first time, you need to wait about 10-20 minutes.

5.3 Phase 3： Adjustment parameters

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

5.3.1 Parameter adjustment logic relationship and parameter adjustment 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 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.

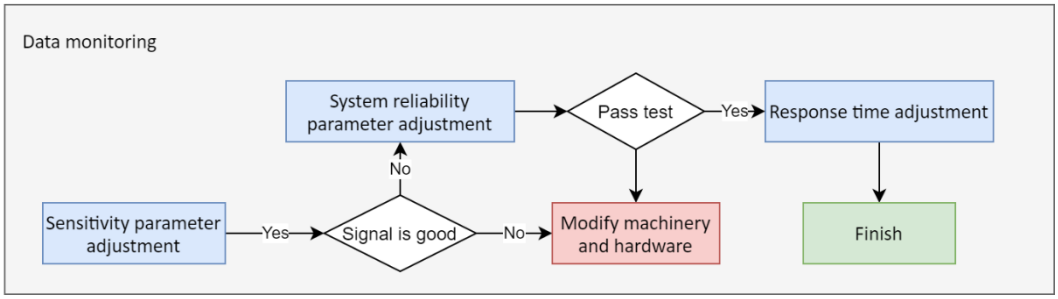


Figure 5-11 Logic block diagram of parameter adjustment

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 will be 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, please plug in and unplug the power again and confirm whether the HID Bridge UART/I2C is connected correctly.

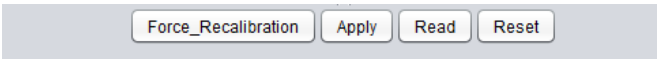


Figure 5-12 Online tuning button

Table 5-4 Description of the buttons for online tuning

Number	Button	Function 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.

5.3.2 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. Since 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.

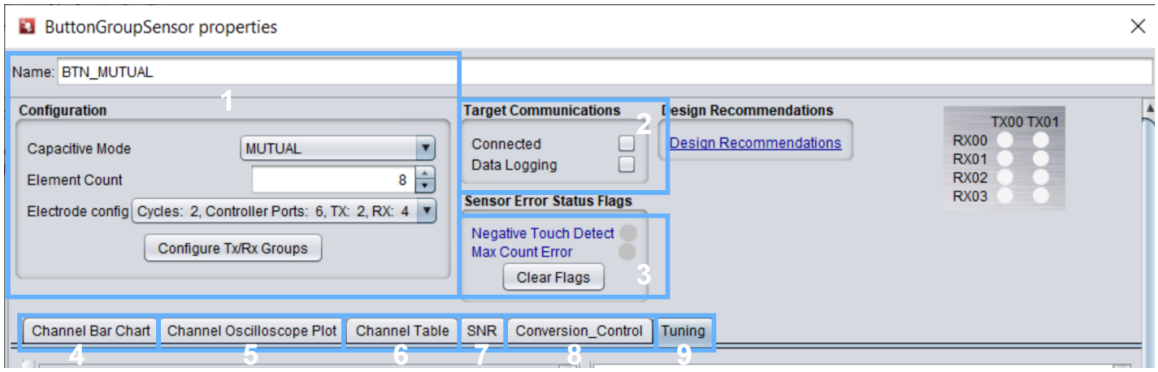


Figure 5-13 Sensor widget interface

Table 5-5 Sensor data monitoring function table

Box in picture	Operation directory	Function	Comment
No. 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.
No. 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.
No. 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.
No. 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.
No. 7	SNR	Multiple sampling to evaluate the rationality of the entire system design and threshold setting.	

5.3.3 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 will amplify 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. Since a larger gain will increase 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 parameter adjustment table

Box in picture	Operation directory	Operation content	Recommended value	Comment
No. 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.
No. 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.

5.3.4 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 parameter adjustment table

Box in picture	Operation directory	Operation content	Recommended value	Comment
No. 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.
No. 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	It is not recommended to change the size, it will cause unresponsiveness after multiple touches.
	Error Threshold	Set the maximum value of the charge conversion output.	conversion count x 1.25	The user needs to increase the processing logic according to the flag bit bMaxCountError.

Generally speaking, the noise source will come from the noise of the CapTIvate™ module itself, the digital signal inside the MCU, the power supply, the input signal at the capacitive touch pin, etc. 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.

5.3.5 Response speed and power assumption 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 parameter table

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

5.4 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

5.5 Phase 5： Develop custom applications

Developing custom applications means understanding the code composition of CapTIvateTM. This part will focus on common custom requirements.

5.5.1 Program structure

First, the composition of the program will be shown, as shown in Figure 5-14. For more related instructions, please refer to Starting from Scratch with the Starter Project.

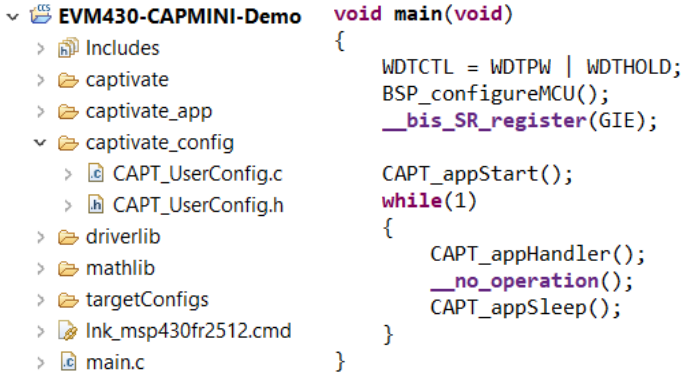


Figure 5-14 Program architecture and main function

The directory structure of the software project is shown in the left figure 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 CapTIvate_config folder. As shown in figure 5-15, the corresponding relationship between the GUI and the parameters in CapTIvate_config

can be clicked on the corresponding parameter in the GUI and read "Affected Software Parameters" for understanding.

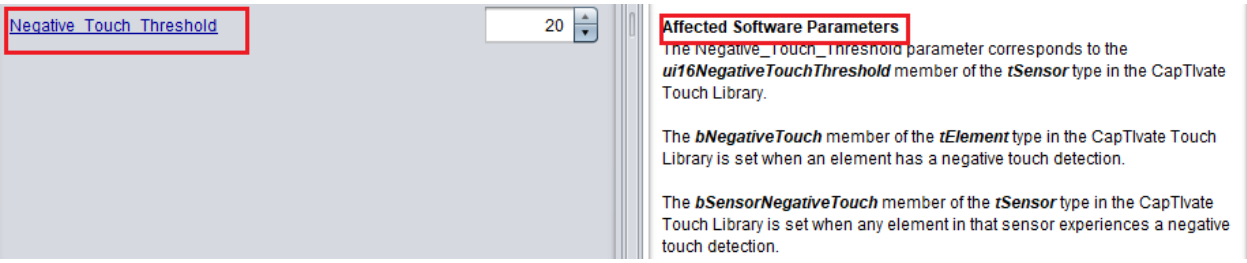


Figure 5-15 Description of the correspondence between GUI and code parameters

Table 5-9 Composition table of software engineering

Number	Contents	Comment
1	CapTIvate™	Low-level driver library for capacitive touch-related peripherals, including capacitance detection and communication configuration.
2	CapTIvate™_app	The top-level logic code of capacitive touch, including system initialization, scanning and calibration logic.
3	CapTIvate™_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	lnk_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 in the right figure of Figure 5-14, and the function functions involved in this part are shown in Table 5-10. For more function descriptions, please refer to API Guide （C:\ti\msp\CapTIvateDesignCenter_x.xx.xx.xx\CapTIvateDesignCenter\docs\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.

5.5.2 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 hung under the Sensor structure, and the Element structure is hung under the Cycle structure. Users can directly access the structure declared in captivate_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 Table of commonly used variables in the Sensor structure

Number	Variable	Types	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 handle the state change of the Sensor during scanning, and it will be 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 Table of commonly used variables in the Element structure

Number	Variable	Types	Comment
1	. bProx	Bool	The Element has detected proximity sensor.
2	. bTouch	Bool	Indicate whether 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, please refer to Register a callback function★. For examples of custom projects, please refer to the CCS/IAR routines corresponding to EVM430-CAPMINI: EVM430-CAPMINI-Demo★。

5.5.3 Realization and customization of communication function

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 GUI will complete the IO 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 captivate_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:
    - Step 1: Modify the CAPT_I2CRegisterReceiveHandler() function in captivate->COMM->CAPT_Interface.c. This function will be 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

    - Step 2: Write a custom frame processing function. As shown in Figure 5-17, MSP430 will read 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 will not reset it. In addition, the receiving/sending data segment in the buffer will have 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

    - Step 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.

```c
#include <msp430.h>
#include "driverlib.h"
#include "captivate.h"
#include "CAPT_App.h"
#include "CAPT_BSP.h"
#include "applications.h"
#include "I2CSlave.h"

#define CUSTOM_KEY_STATE_CMD  0x0B       //I2C CMD Byte

void BTNEventHandler(tSensor *pButton)
{
    if((pButton->bSensorTouch == 1) &&
            (pButton->bSensorPrevTouch == 0))
    {
        I2CSlave_setRequestFlag();
    }
}

void Cus_CAPT_CommunicationPackageAnalysis(uint8_t * pBuffer)
{
    switch (pBuffer[3])
    {
        case CUSTOM_KEY_STATE_CMD:
            pBuffer[3] = BTN00.bSensorTouch;
            break;
        default:
            break;
    }
}

void Panel_Init()
{
    //Customized initialization
    BTN00.pvCallback = (pvCallbackAddress)&BTNEventHandler;

}

void main(void)
{
    WDTCTL = WDTPW | WDTHOLD;
    BSP_configureMCU();
    Panel_Init();
    CAPT_appStart();

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

Figure 5-18 Schematic diagram of the overall code

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

Table 5-13 Customized communication function table

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 CapTIvate module enables the Timer peripheral	Modify the assignment of FUNCTIONTIMER__ENABLE in CapTIvate->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 will reset I2C if a single frame transmission exceeds 8ms.
5	Modify I2C address	Modify I2CSLAVE__ADDRES in CapTIvate->COMM->I2CSlave_Definitions.h	

5.5.4 Bootloader

Bootloader is mainly used for online upgrade. The hardware needs to occupy 4 Pin feet. 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.

User Guide：

MSP430 FRAM Devices Bootloader (BSL) User's Guide

Application Guide:

MSP430 Bootloader With SimpleLink MCUs

MSP430 Bootloader With Sitara Embedded Linux Host

5.5.5 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 burning software is Uniflash and the burning 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 following this link.

- Product page: UniFlash
- User's guide:
  - UniFlash Quick Start Guide
  - Programming the Bootloader of MSP430™ Using UniFlash

MSP-FET★: It is 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★: It is a low-cost MSP430 debug probe and usually sold with LaunchPad. Besides, it only supports a fixed voltage power supply.

- Product page: Please refer to the specific Launchpad product page, such as MSP430FR2311 LaunchPad kit.

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

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

- Product page: MSP-GANG Production Programmer

6 Appendix

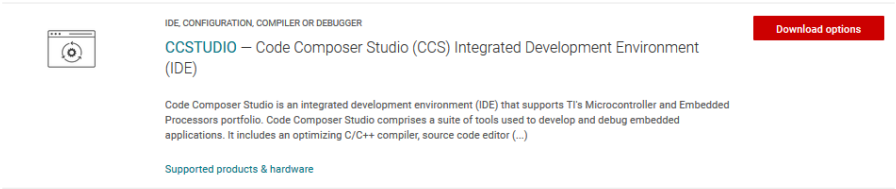
CapTIvate™ Design Center is for code generation and online reference. Code Composer Studio (CCS) is for downloading GUI-generated code and embedded software development. This appendix shows user how to install CCS and the CapTIvate™ Design Center step-by-step, how to quickly evaluate the EVM board provided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

6.1 CCS installation

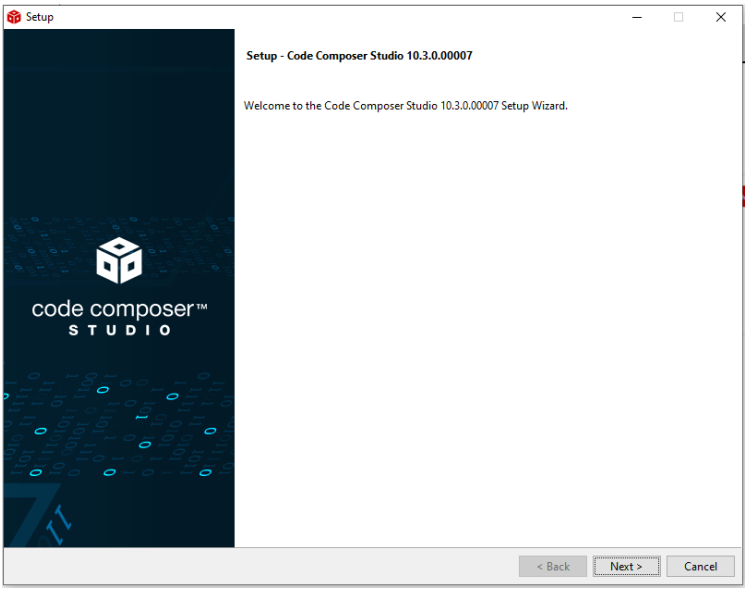
1. **Install Code Composer Studio (CCS) Integrated Development Environment (IDE)**

Project page: [CCS IDE for MSP430]

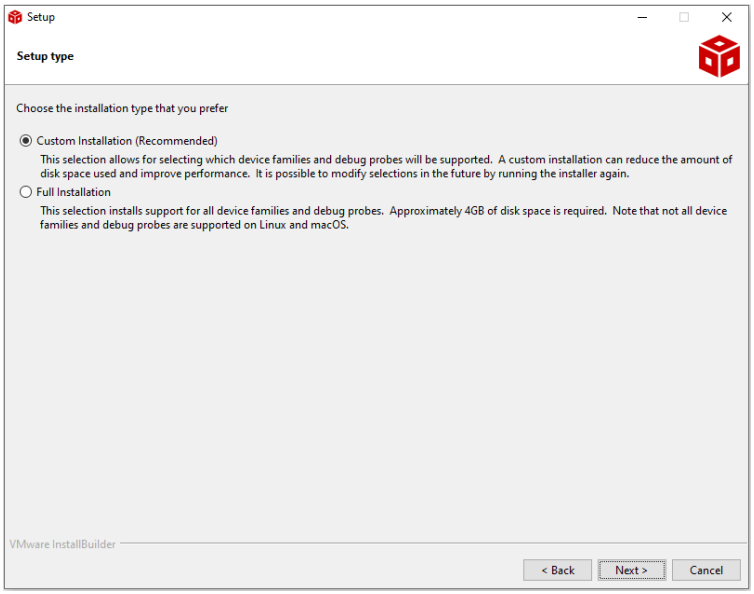
2. **Select the single file installer or the on-demand installer for CCS IDE according to your PC operation system**



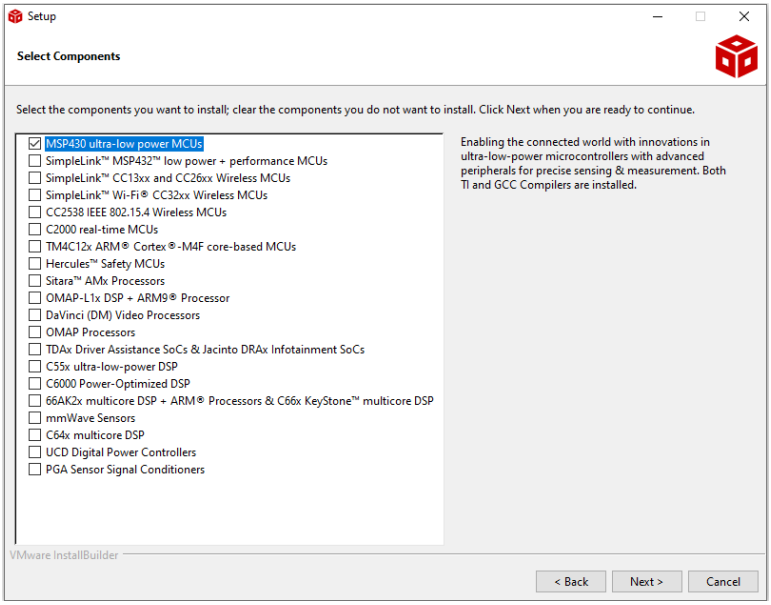
3. **Start CCS IDE installation**



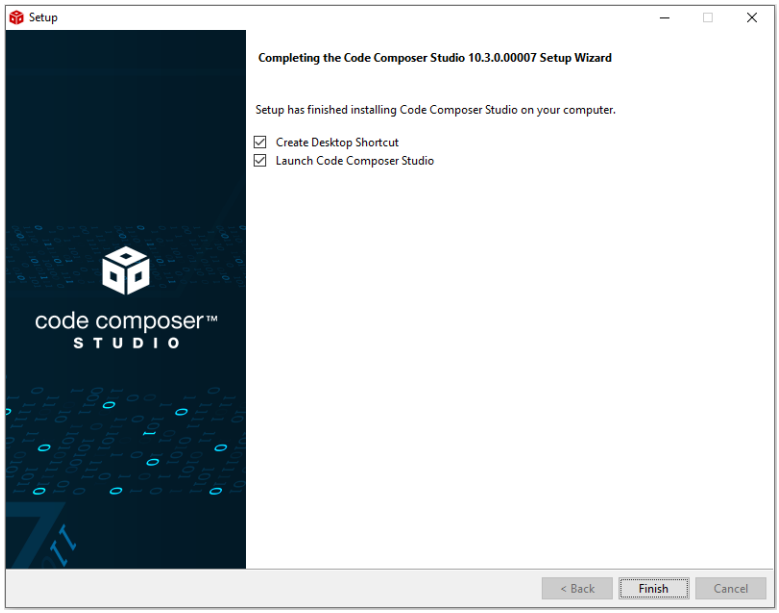
4. **Select “Custom installation” is recommended**



5. **Select "MSP430 ultra-low power MCUs"**



6. **The installation is completed**

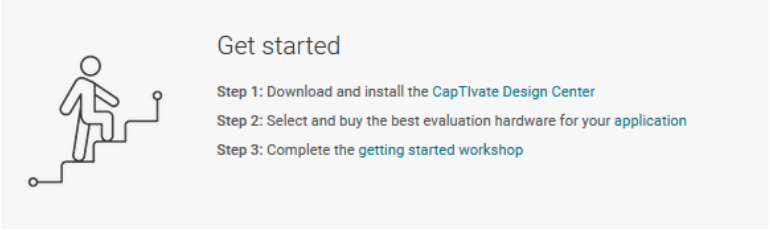


6.2 CapTIvate™ Design Center GUI installation

Note: Please install corresponding Java JDK before CapTIvate™ Design Center installation

1. Install CapTIvate™ Design Center

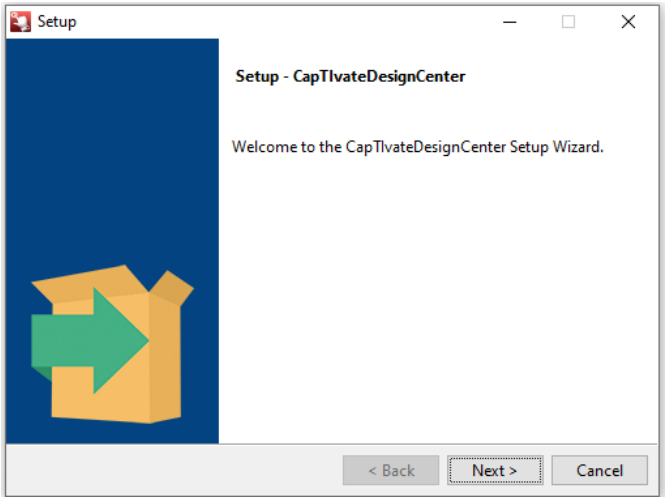
Product page: CapTIvate™ Design Center



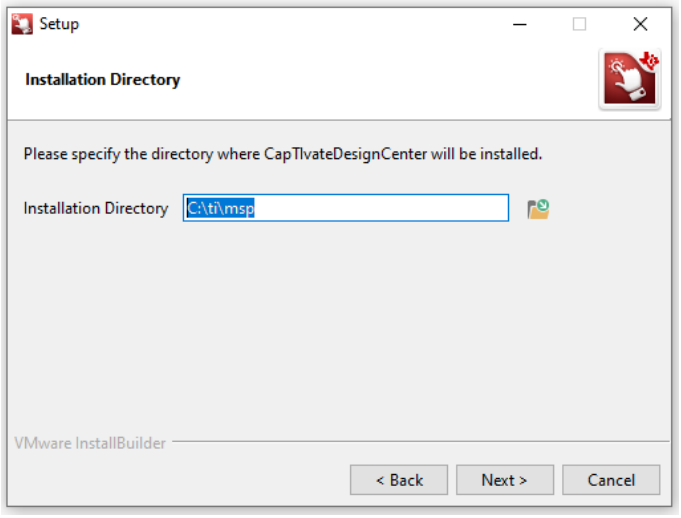
CapTIvate_Design_Center Product downloads

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	MD5 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

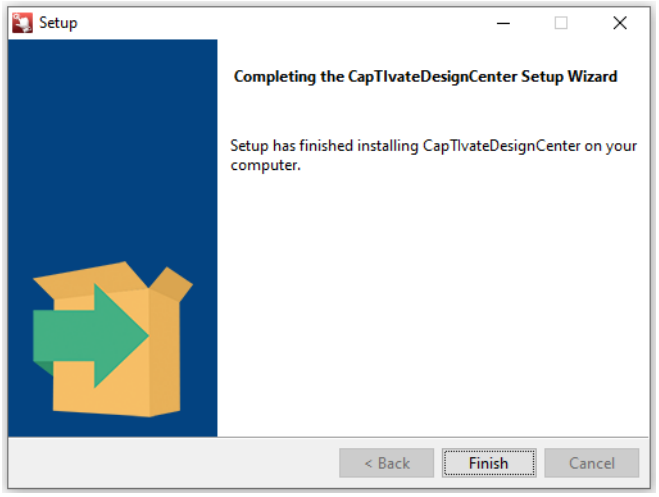
2. Start installation



3. **Keeping the default installation directory is recommended**



4. **Installation is completed**

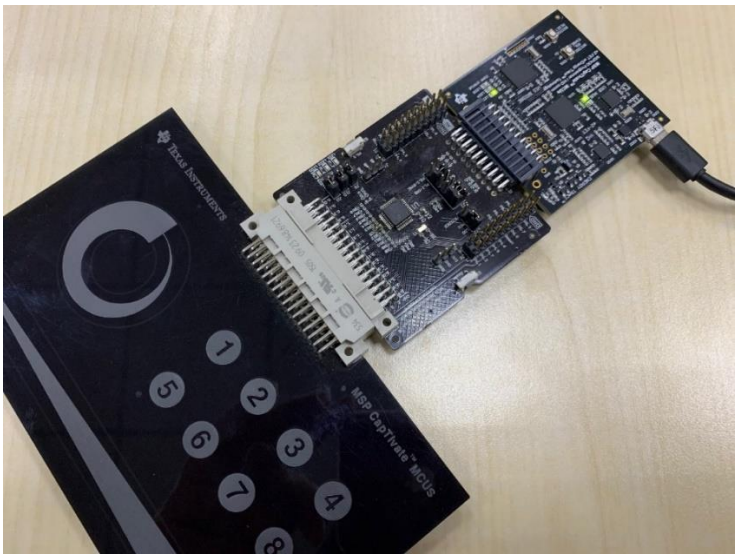


6.3 Hardware connection

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

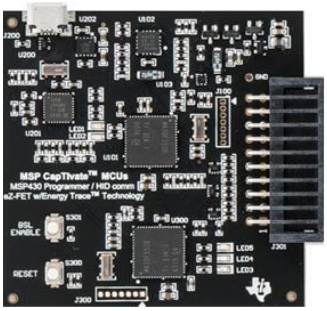
The hardware connection, rapid evaluation, rapid development demonstration will be based on the development chain as the following table. For other EVM boards, the development and evaluation are similar

GUI	IDE	SDK	Programmer	MCU board	Sensor board	Supported Sensor type
CapTIvate™ Design Center	CCS /IAR	MSPWare	CAPTIVATE™-PGMR	CAPTIVATE™-FR2676	CAPTIVATE™-BSWP	Self-mode button/Wheel/slider/Proximity
				CAPTIVATE™-FR2633	CAPTIVATE™-PHONE	Mutual-mode button/Wheel/slider/Proximity
				BOOSTXL-CAPKEYPAD(FR2522)		12 mutual mode buttons and 1 proximity
				EVM430-CAPMINI (FR2512)		4 self-mode buttons



2. CAPTIVATE-PGMR programmer

Programmer product page



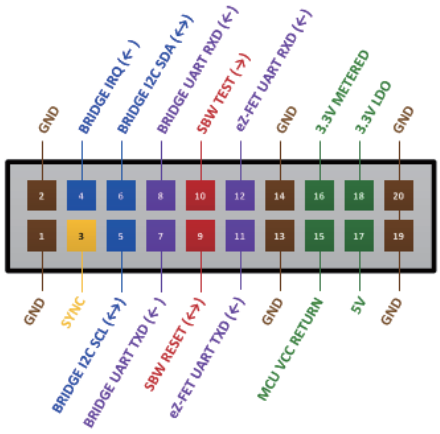
CAPTIVATE-PGMR programmer interface definition

Power pins: 3.3V METERED and 3.3V LDO. Notes: 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.

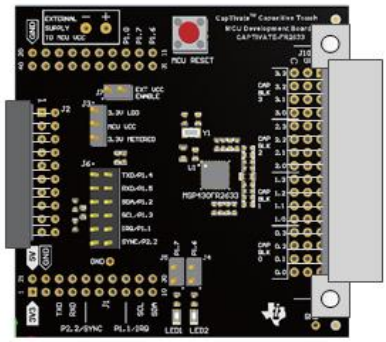
Captivate touch online parameter tuning pins: BRIDGE UART RXD and BRIDGE UART TXD, BRIDGE I2C SDA and BRIDGE I2C SCL.



PCB 20-PIN Male Connector

3. CAPTIVATE-FR2633 MCU board

MCU board product page



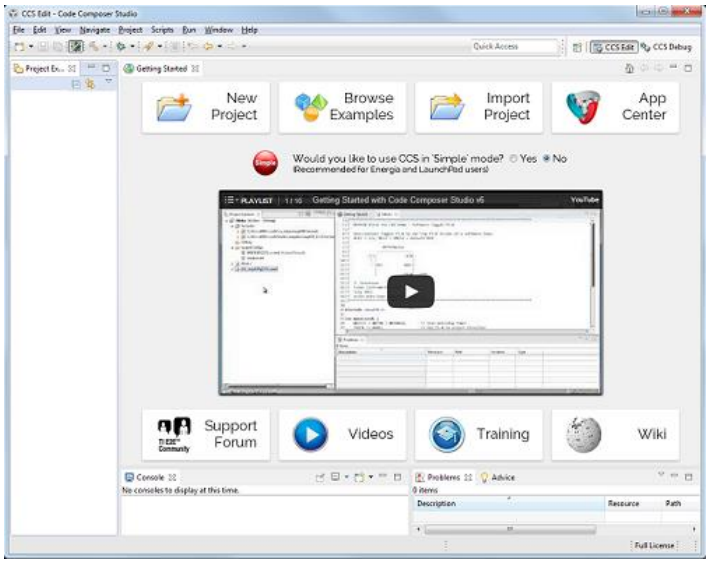
4. CAPTIVATE-BSWP Capacitive touch self-capacitance button, slider, wheel, and proximity sensor demonstration board

Capacitive touch demonstration board product page

6.4 Rapid evaluation

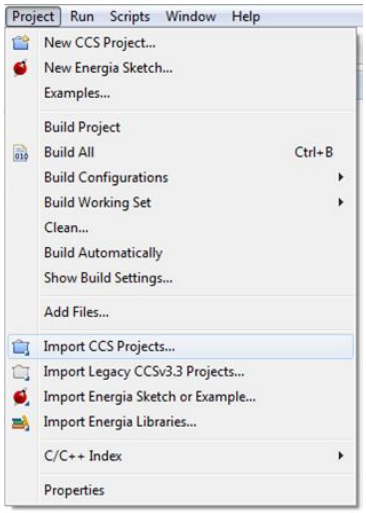
When the CapTIvate™ Design Center is installed and the hardware validation platform is ready, we can import and open the example project insides of CapTIvate™ Design Center on CCS to evaluate the capacitive touch simply and rapidly.

1. **Open CCS**



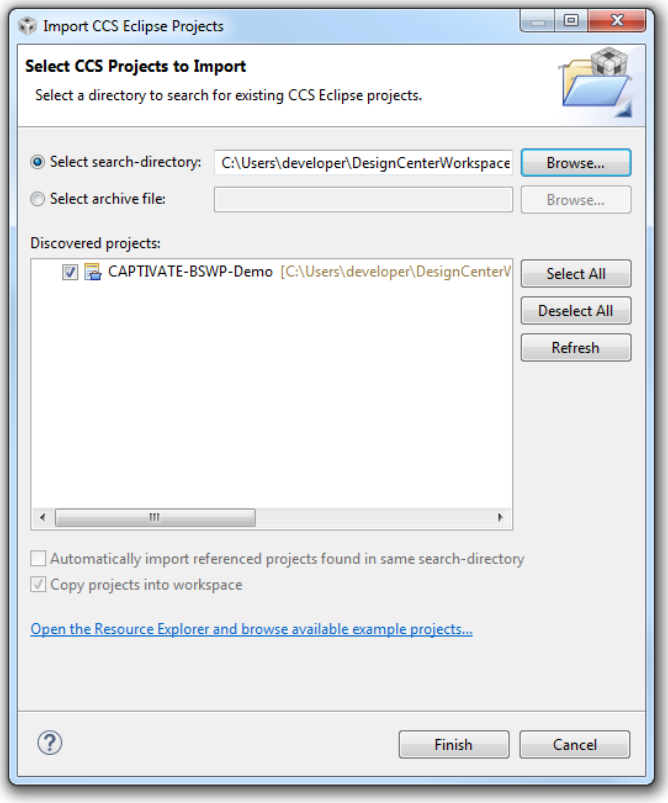
2. **Import CCS project**

Select “Project->Import CCS Projects”



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

The direction is：
C:\Users\Username\CapTIvateDesignCenter_x_xx_xx_xx\CapTIvateDesignCenterWorkspace\TI_Examples\FR2676_CAPTIVATE-BSWP



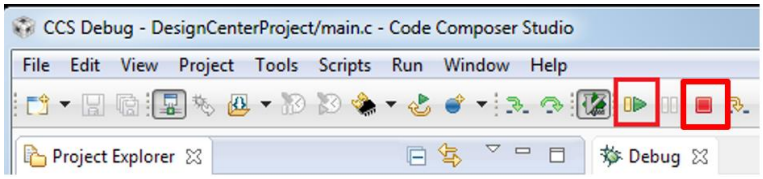
4. Try-tun and evaluate the capacitive touch example project

Click debug button on CCS IDE

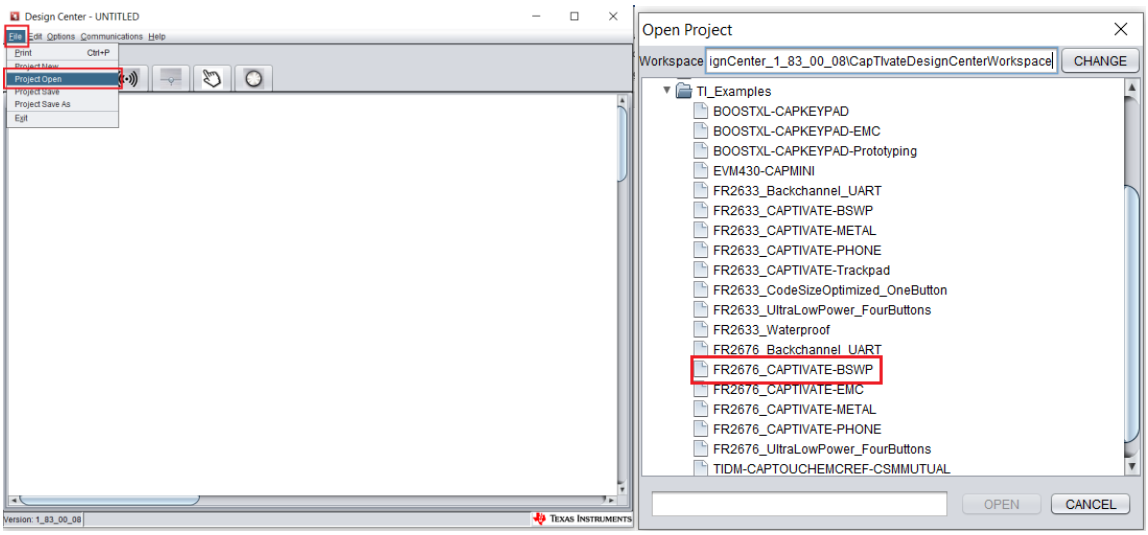
It may take a long time on the first compilation, so please be patient.

```
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.
```

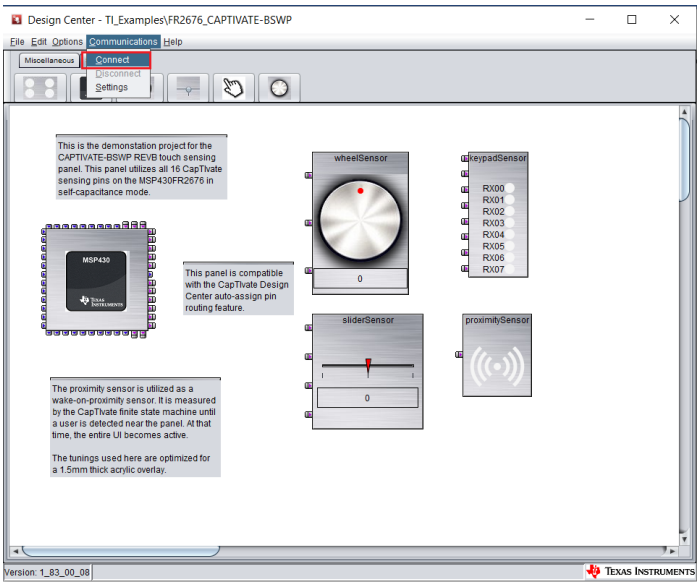
Click the run button and stop button. Then you may need to plug the power supply in again.



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

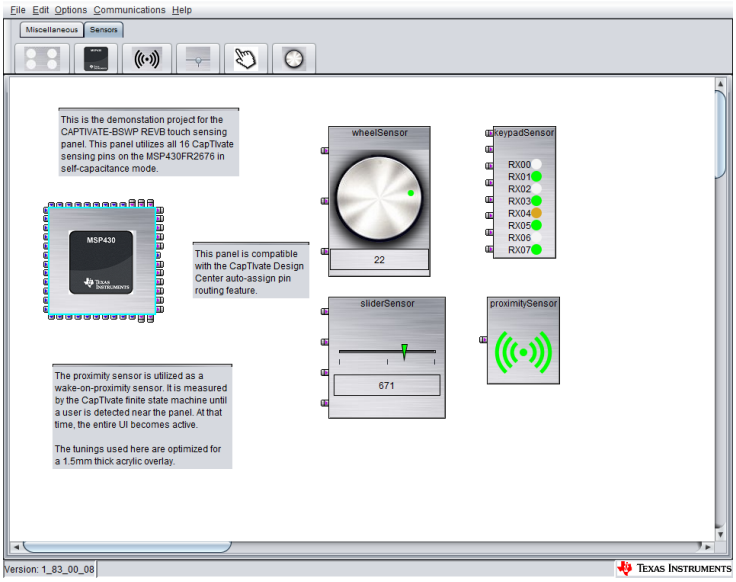


6. Connect CapTIvate™ Design Center with the hardware board system



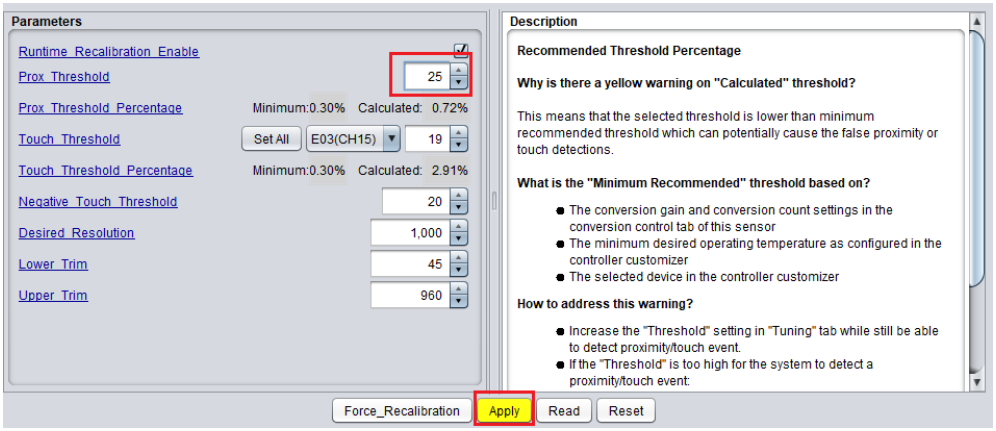
7. Evaluate the capacitive touch features

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



8. Online parameter tuning

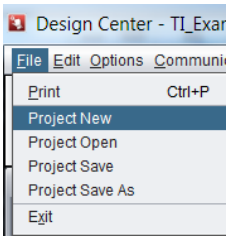
Modify the parameters and click the apply button to complete the online parameters tuning, please see chapter 5.3 of this manual for further instructions and descriptions.



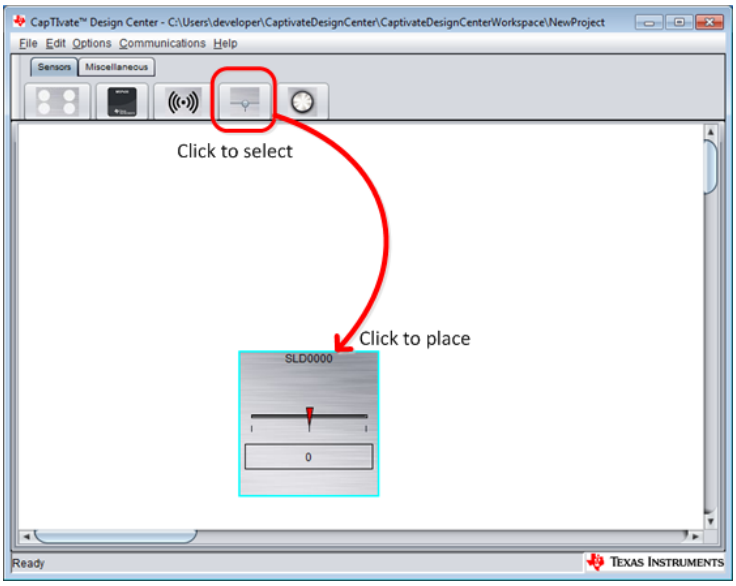
6.5 Rapid development

Before users develop their own hardware board, users can rapidly create the project, configure touch capabilities parameter, and start validation or simple development based on the MSP hardware validation platform and CapTIvate™ Design Center.

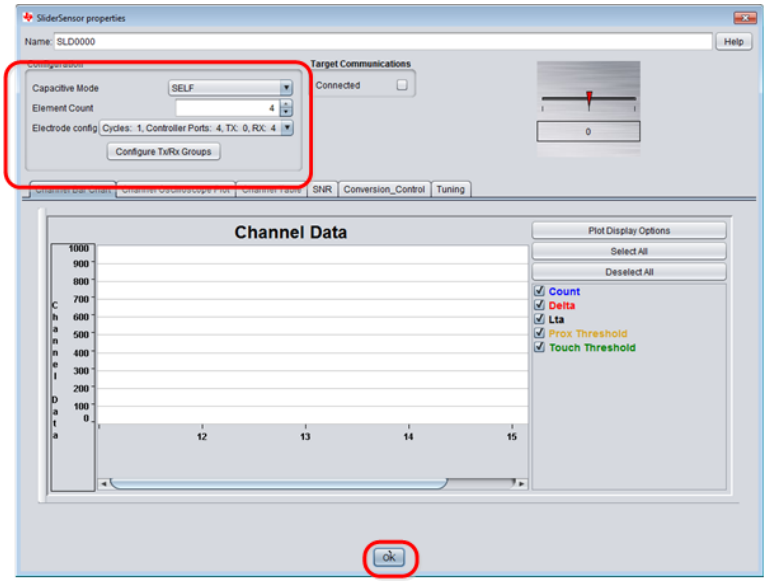
1. Create the new project on CapTIvate™ Design Center



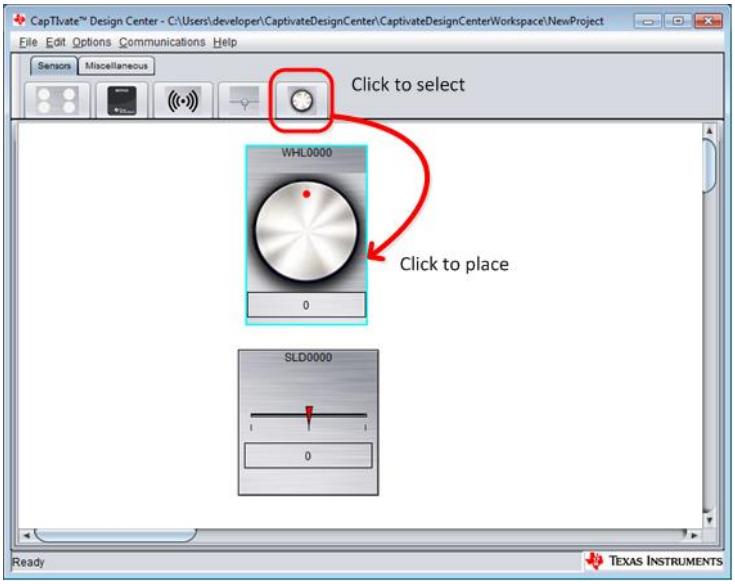
2. Place a slider sensor in the design area



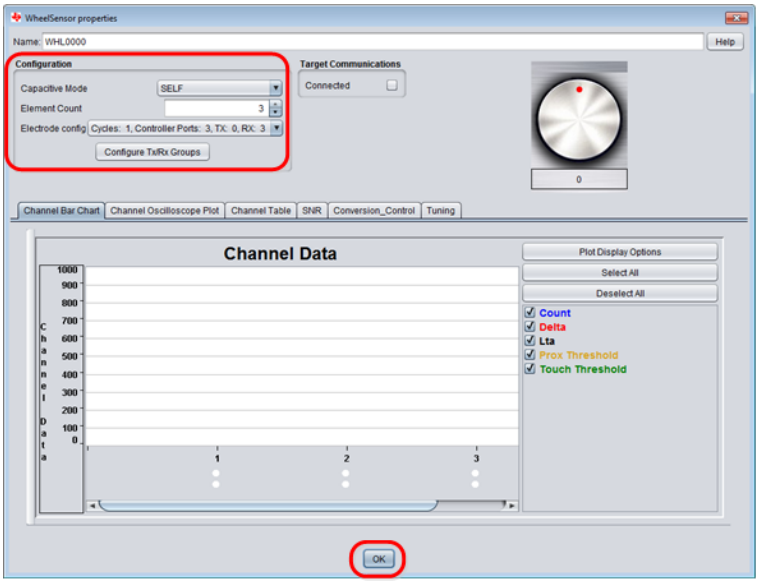
3. **Configure the slider sensor on properties dialog**



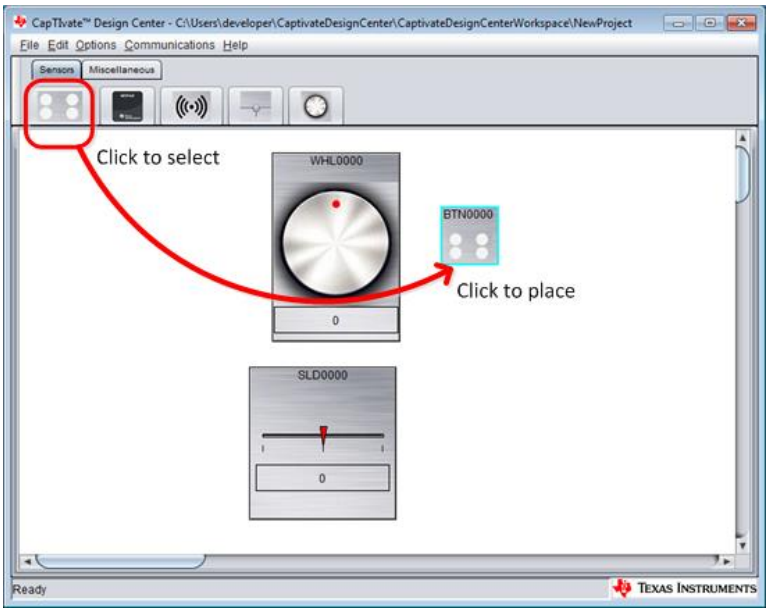
4. **Place a wheel sensor in the design area**



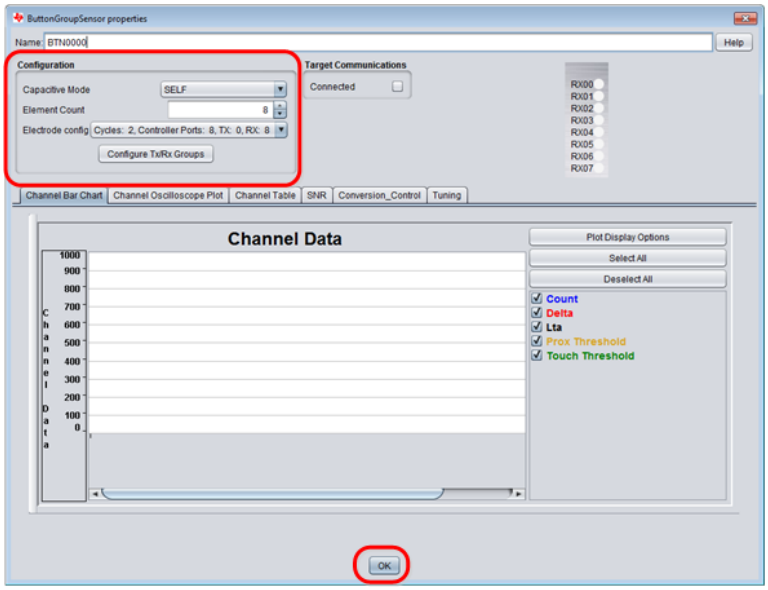
5. **Configure the wheel sensor on properties dialog**



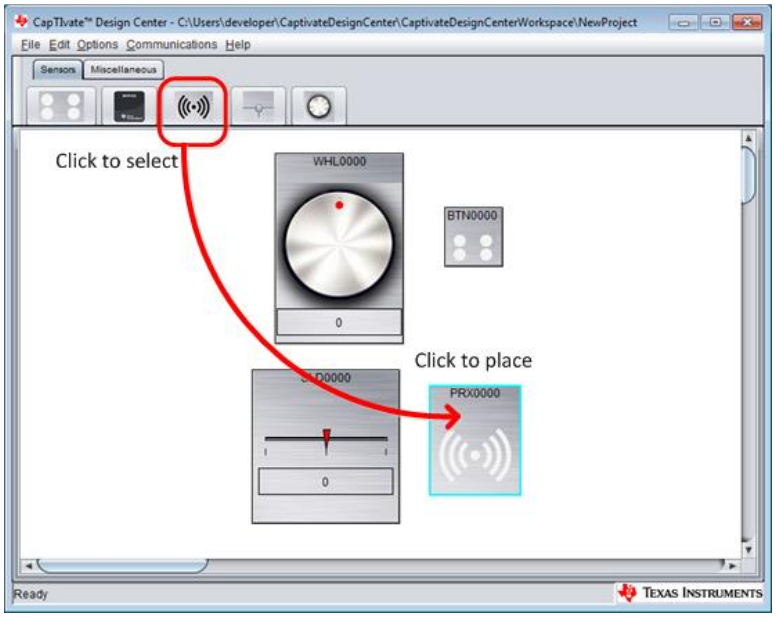
6. **Place a button group (keypad) sensor in the design area**



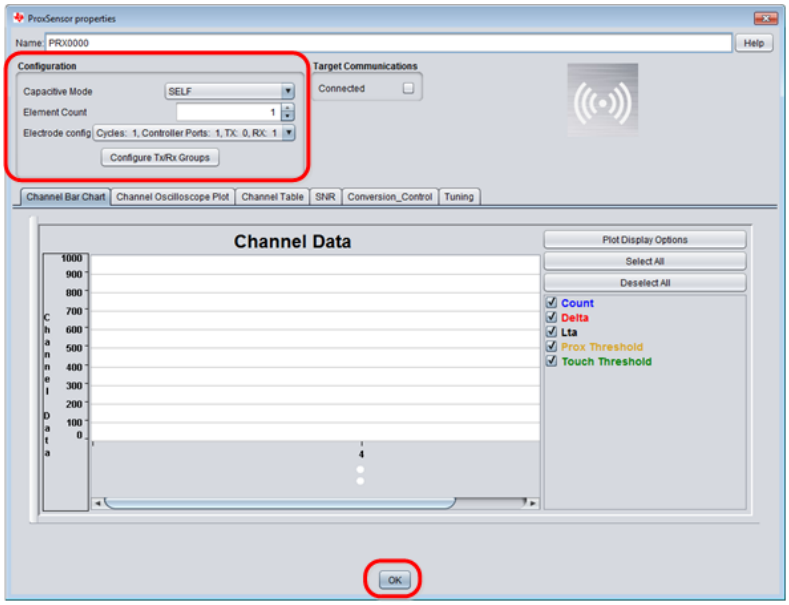
7. Configure the button group sensor on properties dialog



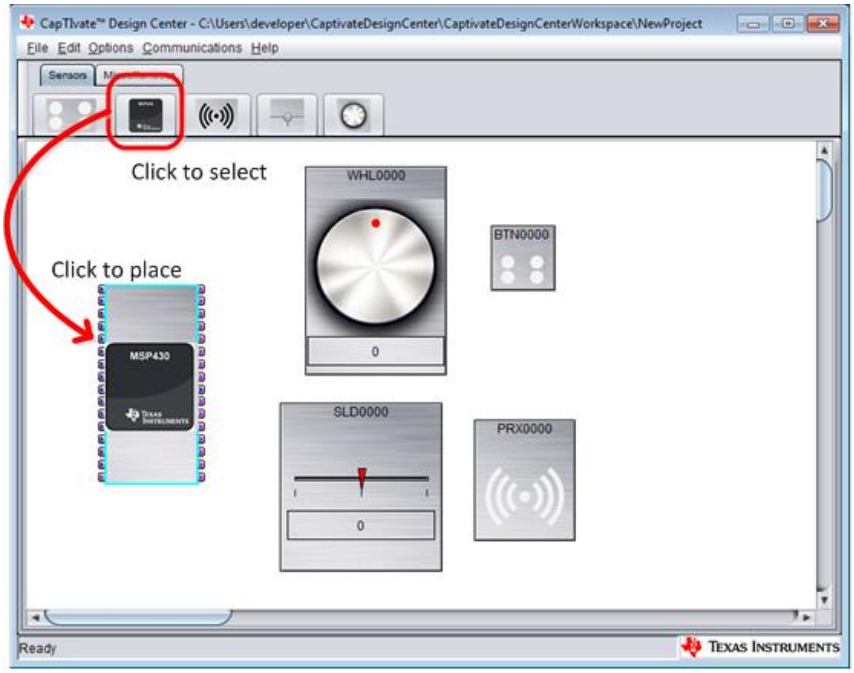
8. Place a proximity sensor in the design area



9. Configure the proximity sensor on properties dialog



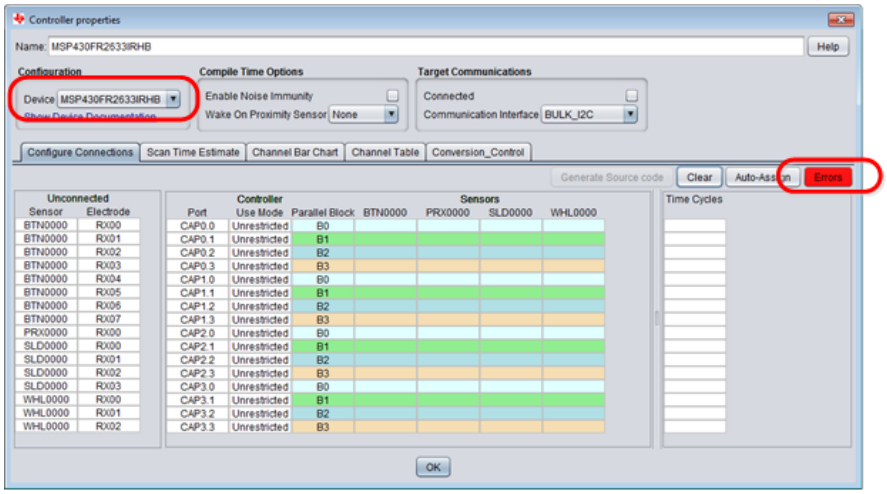
10. Select and place the MSP430 controller

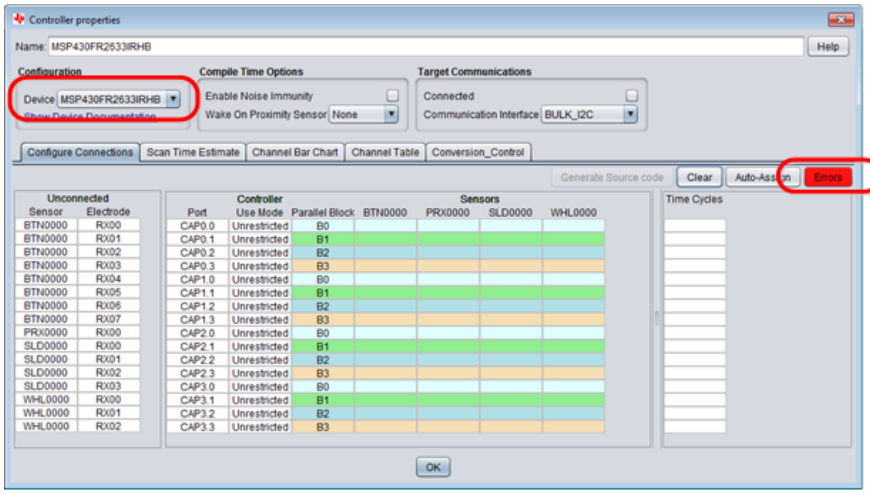


11. Connect sensors to MSP430 capacitive touch I/O ports

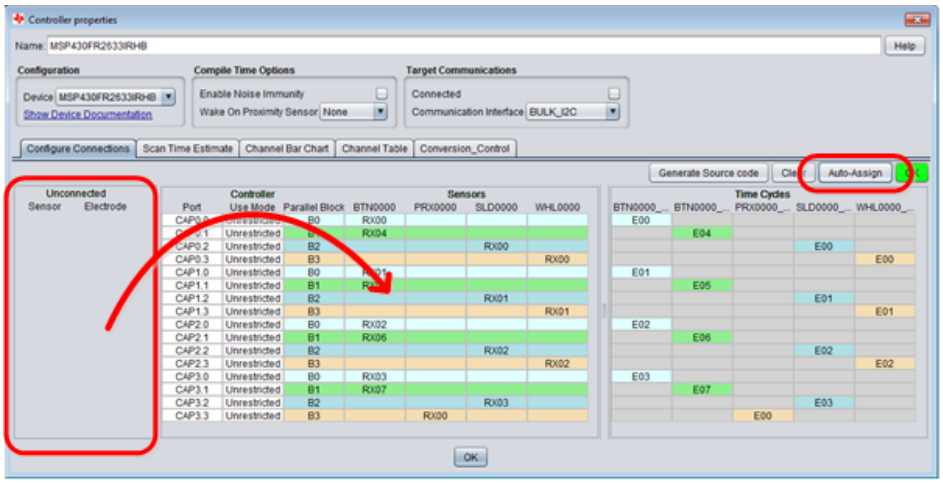
Double-click on the MSP430 controller object in the design area to display its properties. Configure the MSP430 controller as MSP430FR2633IRHB (32-pin QFN package)

Note that the "Errors" LED is red, indicating that there are still unconnected sensor ports.

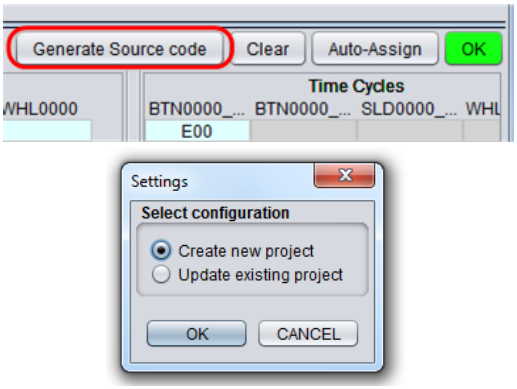




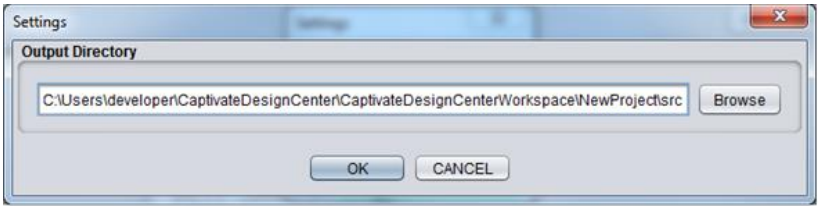
Select the “Auto-Assign” button to automatically assign all the sensor ports to appropriate ports on the MSP430. Note that the “Errors” LED turns green and “OK”, indicating that all sensor ports have been assigned to controller ports.



12. Generate source code

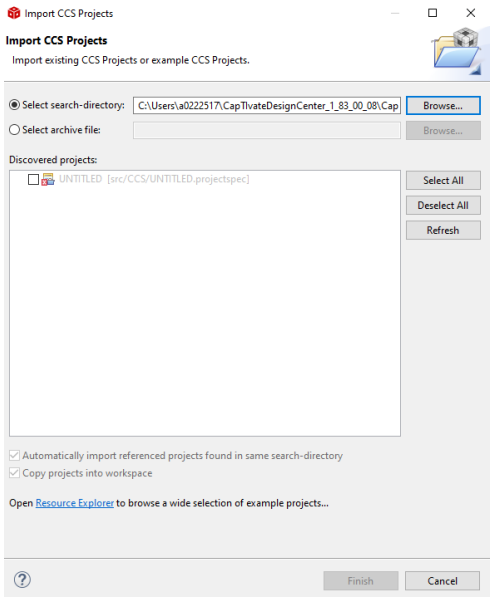


Saving the source code on default location of CapTIvate™ Design Center is recommended.



13. Import the project on CCS, Start development

After completed the rapid evaluation process according to chapter 6.4, the project generated after adjusting the parameters on CapTIvate™ 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.



Note: for saving code size, code compilation optimization selects the highest level. It is recommended that Optimization be turned off during actual development.