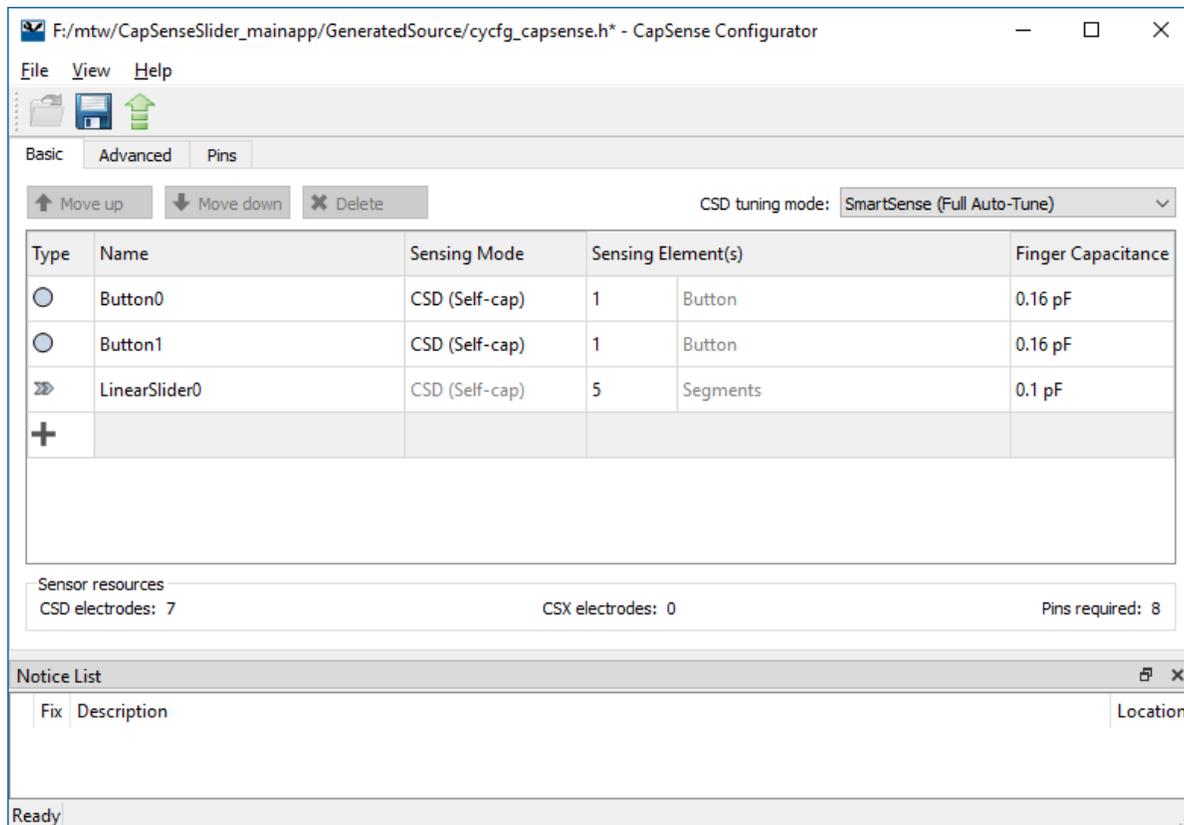


Overview

CapSense is a Cypress capacitive sensing solution. Capacitive sensing can be used in a variety of applications and products where conventional mechanical buttons can be replaced with sleek human interfaces to transform the way users interact with electronic systems. These include home appliances, automotive, IoT, and industrial applications. CapSense supports multiple interfaces (widgets) using both CSX and CSD sensing methods, with robust performance.

The CapSense Configurator is part of a collection of tools included in ModusToolbox. Use it to create and configure CapSense widgets, and generate code to control the application firmware. There is a separate CapSense Tuner application for tuning, testing, and debugging for easy and smooth design of human interfaces on customer products.



The screenshot shows the CapSense Configurator window with the following components:

- File menu: File, View, Help
- Navigation: Move up, Move down, Delete
- CSD tuning mode: SmartSense (Full Auto-Tune)
- Widget Table:

Type	Name	Sensing Mode	Sensing Element(s)		Finger Capacitance
<input type="radio"/>	Button0	CSD (Self-cap)	1	Button	0.16 pF
<input type="radio"/>	Button1	CSD (Self-cap)	1	Button	0.16 pF
<input checked="" type="radio"/>	LinearSlider0	CSD (Self-cap)	5	Segments	0.1 pF
+					

Sensor resources:

- CSD electrodes: 7
- CSX electrodes: 0
- Pins required: 8

Notice List:

Fix	Description	Location

Ready

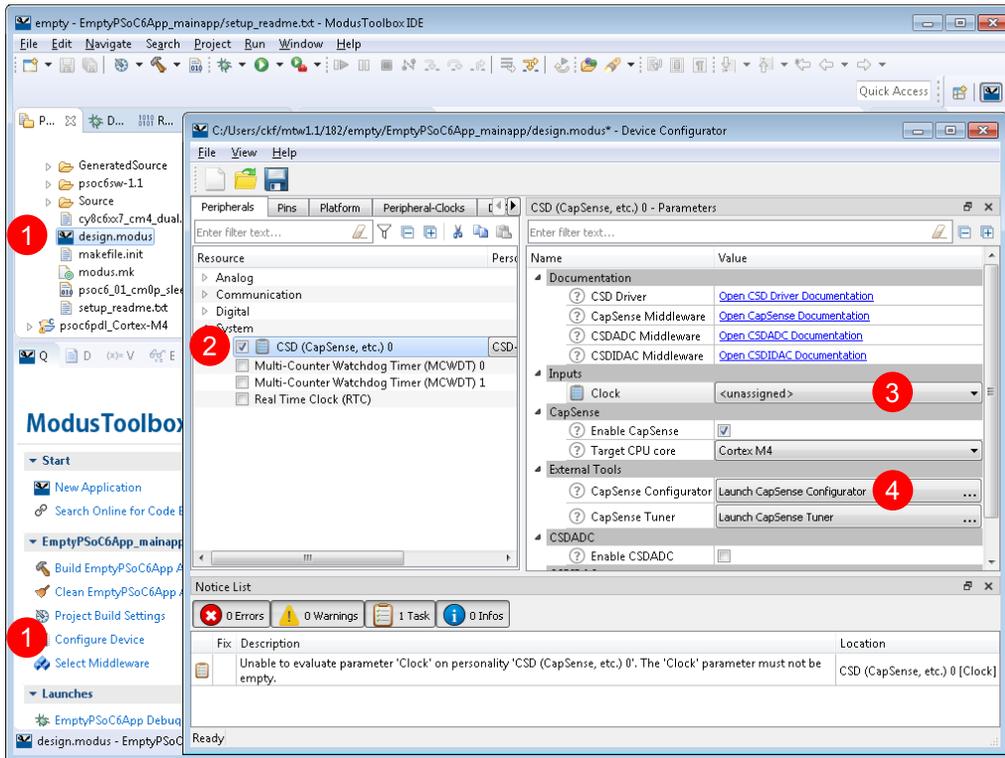
Launch the CapSense Configurator

The CapSense Configurator is a stand-alone tool that contains [menus](#) and [tabs](#) to configure CapSense settings. You can run the configurator from, and use it with, a ModusToolbox IDE application. You can also run it independently of the IDE. Then, you can either use the generated source with a ModusToolbox IDE application, or use it in any software environment you choose.

From a ModusToolbox IDE Application

1. Launch the Device Configurator from the ModusToolbox IDE.
2. On the **Peripherals** tab, enable the **CSD (CapSense)** resource.

3. On the **Parameters** pane, select an appropriate input **Clock**.
4. On the **Parameters** pane, click the **Launch CapSense Configurator** button.



The Device Configurator is based on the *design.modus* file, which contains all the required information about the device and the application. This information is passed to the CapSense Configurator. When you save changes, it generates/updates firmware in the ModusToolbox IDE application's "GeneratedSource" folder.

Independent of the ModusToolbox IDE

To run the CapSense Configurator independently, navigate to the install location and run the executable. The default install location for the CapSense Configurator is:

```
<install_dir>/tools/capsense-configurator-<version>
```

When run independently, the configurator opens without any information. You must open a header file for the application for which you want to configure CapSense. If a header file does not exist, then it will be created when clicking the **Save** button.

From the Command Line

You can run the configurator from the command line. However, there are only a few reasons to do this in practice. The primary use case would be to re-generate source code based on the latest configuration settings. This would often be part of an overall build script for the entire application.

For information about command line options, run the configurator using the `-h` option.

Quick Start

This section provides a simple workflow for how to use the CapSense Configurator.

1. Create a CapSense application using the ModusToolbox IDE. It provides a simple CapSense Linear Slider example to get started.
2. Launch the Device Configurator.
3. Enable and configure a communication peripheral. The example uses and SCB configured as EZI2C.
4. Review the selected input clock and change as needed.
5. [Launch the CapSense Configurator](#).
6. Add and configure widgets on the [Basic tab](#).
7. Configure parameters on the [Advanced tab](#).
8. Assign pins on the [Pins tab](#).
9. Save to generate code.

The CapSense Configurator generates code into a “GeneratedSource” directory in your ModusToolbox IDE application, or in the location you specified for non-ModusToolbox IDE applications. That directory contains the necessary source (.c) and header (.h) files for the generated firmware, which uses the relevant driver APIs to configure the hardware.

10. Include `cycfg_capsense.h` in the `main.c` file.
11. Build the application in the ModusToolbox IDE, and program the device.
12. Launch the CapSense Tuner. Refer to the *CapSense Tuner Guide*.

Menus/Commands

The menus include:

- **File** – Provides basic commands to open, close, and save files, as well as exit the configurator.
 - **Open** – Open an existing `cycfg_capsense.h` file. The current file, if any, will be closed.
 - **Save** – Save the current file and generate CapSense middleware configuration code. If there are errors in the application, a dialog will indicate such. The file will still be saved.
 - **Export** – Export the current configuration register map in PDF format.
 - **Exit** – Close the configurator. You will be prompted to save any pending changes.
- **View** – This menu contains a command to hide or show the Notice List pane. The pane is shown by default. There is also a command to show or hide the toolbar.
- **Help** – Provides access to this document and an About box.

Notice List

The Notice List pane combines notices (errors, warnings, tasks, and notes) from many places in the configuration into a centralized list. If a notice shows a location, you can double-click the entry to show the error or warning.

Notice List			✕
Fix	Description	Location	
	The multi-frequency scan is incompatible with SmartSense	CapSense Configurator	

The Notice List pane contains the following columns:

- **Icon** – Displays the icons for the error, warning, task, or note.
- **Fix** – This may display a wrench icon, which can be used to automatically address the required notice.
- **Description** – Displays a brief description of the notice.
- **Location** – Displays the specific tab of the message, when applicable.

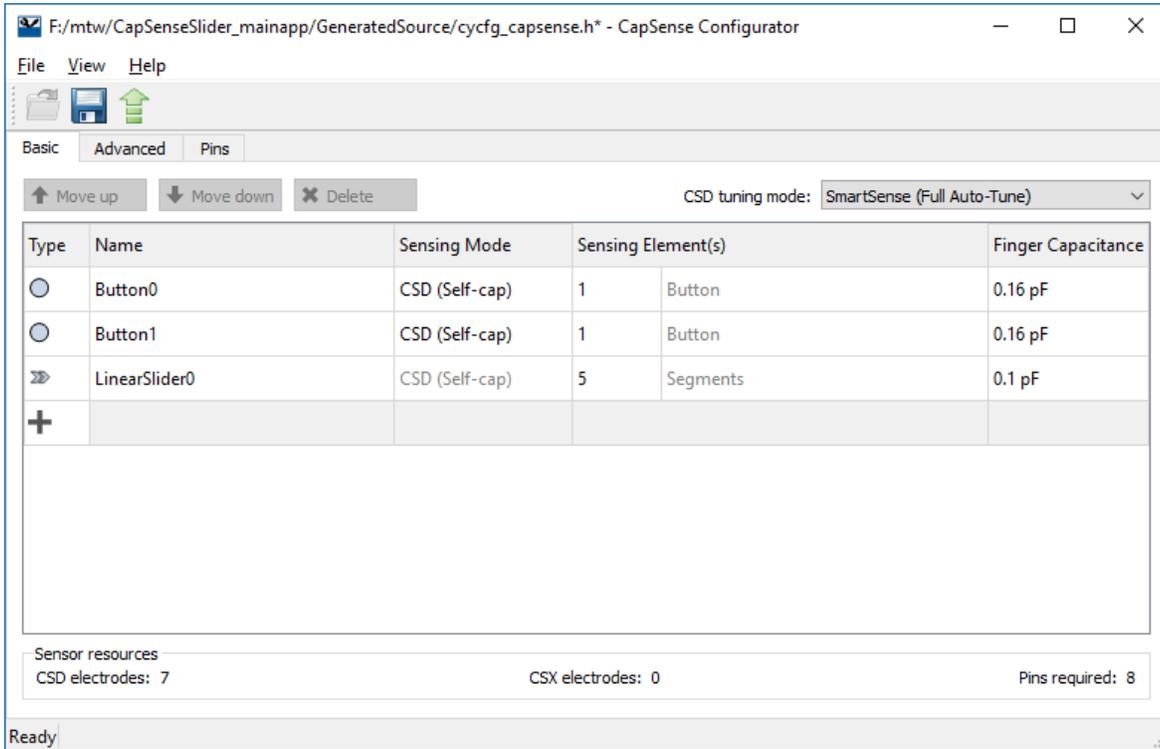
Tabs

The CapSense Configurator contains the following tabs, each of which provides access to specific parameters. Separate sections in this document provide more descriptions of these tabs.

- [Basic Tab](#)
- [Advanced Tab](#)
- [Pins Tab](#)

Basic Tab

The **Basic** tab defines the high-level middleware configuration. Use this tab to add various *Widget Type* and assign *Sensing mode*, *Widget Sensing Element(s)* and *Finger capacitance* for each widget.



Move up Move down Delete CSD tuning mode: SmartSense (Full Auto-Tune)

Type	Name	Sensing Mode	Sensing Element(s)		Finger Capacitance
<input type="radio"/>	Button0	CSD (Self-cap)	1	Button	0.16 pF
<input type="radio"/>	Button1	CSD (Self-cap)	1	Button	0.16 pF
<input checked="" type="radio"/>	LinearSlider0	CSD (Self-cap)	5	Segments	0.1 pF
+					

Sensor resources
 CSD electrodes: 7 CSX electrodes: 0 Pins required: 8

Ready

The following table provides descriptions of the various **Basic** tab parameters:

Name	Description
CSD tuning mode	<p>Tuning is a process of finding appropriate values for configurable parameters (Hardware parameters and Threshold parameters) for proper functionality and optimized performance of the CapSense system.</p> <p>The SmartSense Auto-tuning is an algorithm embedded in the CapSense middleware. The algorithm automatically finds optimum values for configurable parameters basing on the hardware properties of capacitive sensors. This allows the user to avoid the manual-tuning process.</p> <p>Configurable parameters that affect the operation of the sensing hardware are called Hardware parameters. Parameters that affect the operation of the touch-detection firmware algorithm are called Threshold parameters.</p> <p>This parameter is a drop-down box to select the tuning mode for CSD widgets only.</p> <ul style="list-style-type: none"> ▪ SmartSense (Full Auto-Tune) – This is the quickest way to tune a design. Most hardware and threshold parameters are automatically tuned by the middleware and the configurator GUI displays them as <i>Set by SmartSense</i> mode. In this mode, the following parameters are automatically tuned: <ul style="list-style-type: none"> ○ <i>CSD Settings</i> tab: <i>Enable IDAC auto-calibration</i>. ○ <i>Widget Details</i> tab: The CSD-related parameters of the <i>Widget Hardware Parameters</i> and <i>Widget Threshold Parameters</i> groups ○ <i>Widget Details</i> tab: the <i>Compensation IDAC value</i> parameter if <i>Enable compensation IDAC</i> is set. ▪ SmartSense (Hardware parameters only) – The Hardware parameters are automatically set by the middleware. The Threshold parameters are set manually by the user. This mode consumes less memory and less CPU processing time, this leads to consuming lower average power. In this mode, the following parameters are automatically tuned: <ul style="list-style-type: none"> ○ <i>CSD Settings</i> tab: <i>Enable IDAC auto-calibration</i>. ○ <i>Widget Details</i> tab: The CSD-related parameters of the <i>Widget Hardware Parameters</i> group. ○ <i>Widget Details</i> tab: <i>Compensation IDAC</i> value parameter if <i>Enable compensation IDAC</i> is set. ▪ Manual – The SmartSense auto-tuning is disabled, the <i>Widget Hardware Parameters</i> and <i>Widget Threshold Parameters</i> are tuned manually. The lowest memory and CPU process-time consumption. <p>The SmartSense Auto-tuning (both Full Auto-Tune and Hardware parameters only) supports the <i>IDAC Sourcing</i> configuration only.</p> <p><i>SmartSense Auto-tuning</i> requires Modulator clock frequency Modulator clock set to 6000 kHz or higher.</p> <p>SmartSense operating conditions:</p> <ul style="list-style-type: none"> ▪ Sensor capacitance C_p range 5 pF to 61 pF ▪ Maximum external series resistance on a sensor $R_{ext} < 1.1$ kOhm

Name	Description
Widget Type	<p>A widget is one sensor or a group of sensors that perform a specific user-interface functionality. The following widgets types consist:</p> <ul style="list-style-type: none"> ▪ Button – One or more sensors. Each sensor in the widget can detect the presence or absence (i.e. only two states) of a finger on the sensor. ▪ Linear Slider – More than one sensor arranged in the specific order to detect the presence and movement of a finger on a linear axis. If a finger is present, Linear Slider detects the physical position (single axis position) of the finger. ▪ Radial Slider – More than one sensor arranged in the circular order to detect the presence and radial movement of a finger. If a finger is present, the Radial Slider detects the physical position of the finger. ▪ Matrix Buttons – Two or more sensors arranged in the specific horizontal and vertical order to detect the presence or absence of a finger on the intersections of vertically and horizontally arranged sensors. If M and N are the numbers of the sensors in the horizontal and vertical axis respectively, the total of the M x N intersection positions can detect a finger touch. When using the <i>CSD sensing method</i>, a simultaneous finger touch on more than one intersection is invalid and produces invalid results. This limitation does not apply when using the <i>CSX sensing method</i> and all intersections can detect a valid touch simultaneously. ▪ Touchpad – Multiple sensors arranged in the specific horizontal and vertical order to detect the presence or absence of a human finger. If a finger is present, the widget will detect the physical position (both X and Y axis position) of the touch. The <i>CSD sensing method</i> supports detection of up to 2 simultaneous touches (when Advanced Centroid is enabled). The <i>CSX sensing method</i> supports detection of up to 3 simultaneous finger touches. ▪ Proximity Sensor – One or more sensors. Each sensor in the widget can detect the proximity of conductive objects, such as a human hand or finger to the sensors. The proximity sensor has two thresholds: <ul style="list-style-type: none"> ○ <i>Proximity threshold</i> – To detect an approaching hand or finger. ○ <i>Touch threshold</i> – To detect a finger touch on the sensor.
Widget Name	<p>A widget name can be defined to aid in referring to a specific widget in a design. A widget name does not affect functionality or performance. A widget name is used throughout source code to generate macro definitions. A maximum of 16 alphanumeric characters (the first letter must be an alphabetic character) is acceptable for a widget name.</p>
Sensing mode	<p>The parameter to select the sensing mode for each widget:</p> <ul style="list-style-type: none"> ▪ CSD sensing method (Capacitive Sigma Delta) – A Cypress patented method of performing self-capacitance measurements. All widget types support the CSD sensing. ▪ CSX sensing method – A Cypress patented method of performing mutual-capacitance measurements; only buttons, matrix buttons, and touchpad widgets support CSX sensing.

Name	Description
Widget Sensing Element(s)	<p>A sensing element refers to the sensing terminals assigned to port pins to connect to physical sensors on a user-interface panel (such as a pad or layer on a PCB, ITO, or FPCB).</p> <p>The following element numbers are supported by the <i>CSD sensing method</i>:</p> <ul style="list-style-type: none"> ▪ <i>Button</i> – Supports 1 to 32 sensors within a widget. ▪ <i>Linear Slider</i> – Supports 3 to 32 segments within a widget. ▪ <i>Radial Slider</i> – Supports 3 to 32 segments within a widget. ▪ <i>Matrix Buttons</i> – Support 2 to 16 rows and columns. ▪ <i>Touchpad</i> – Supports 3 to 16 rows and columns. ▪ <i>Proximity</i> – Supports 1 to 16 sensors within a widget. <p>The following element numbers are supported by the <i>CSX sensing method</i>:</p> <ul style="list-style-type: none"> ▪ <i>Button</i> – 1 to 32 Rx electrodes (for 1 to 32 sensors) and Tx is fixed to 1. ▪ <i>Matrix Buttons</i> – 2 to 16 Tx and Rx. ▪ <i>Touchpad</i> – Supports 3 to 16 Tx and Rx. The total intersections (node) number is equal to Tx x Rx. The maximum number of nodes is 256.
Finger capacitance	<p>Finger capacitance is defined as capacitance introduced by a user touch on the sensors. This parameter is used to indicate how a sensitive CSD widget is tuned by the <i>SmartSense Auto-tuning</i> algorithm.</p> <p>The supported Finger capacitance range:</p> <ul style="list-style-type: none"> ▪ <i>SmartSense (Full Auto-Tune)</i> mode – 0.1 pF to 1 pF. ▪ <i>SmartSense (Hardware parameters only)</i> mode – 0.02 pF to 20.48 pF on the exponential scale. <p>CapSense sensor sensitivity is inversely proportional to a finger capacitance value. A smaller value of finger capacitance provides higher sensitivity for a sensor. To detect a user touch on a thick overlay (4-mm plastic overlay), finger capacitance is set to a small value, e.g. 0.1 pF. For a sensor with a thin overlay or no overlay, the 0.1 pF finger capacitance setting makes the sensor too sensitive and may cause false touches. For the robust operation, it is important to set the appropriate finger capacitance value by considering the sensor size and overlay thickness of the design. Refer to the CapSense design guide for more information.</p>
Move up / Move down	Moves the selected widget up or down by one on the list. It defines the widget scanning order.
Delete	Deletes the selected widget from the list.
CSD electrodes	Indicates the total number of electrodes (port pins) used by the CSD widgets.
CSX electrodes	Indicates the total number of electrodes (port pins) used by the CSX widgets.
Pins required	Indicates the total number of port pins required for the design. This does not include port pins used by other peripherals in the application or SWD pins in Debug mode. Pins required includes the number of CSD and CSX electrodes, Cmod, Csh, Shield, CintA and CintB electrodes.

Advanced Tab

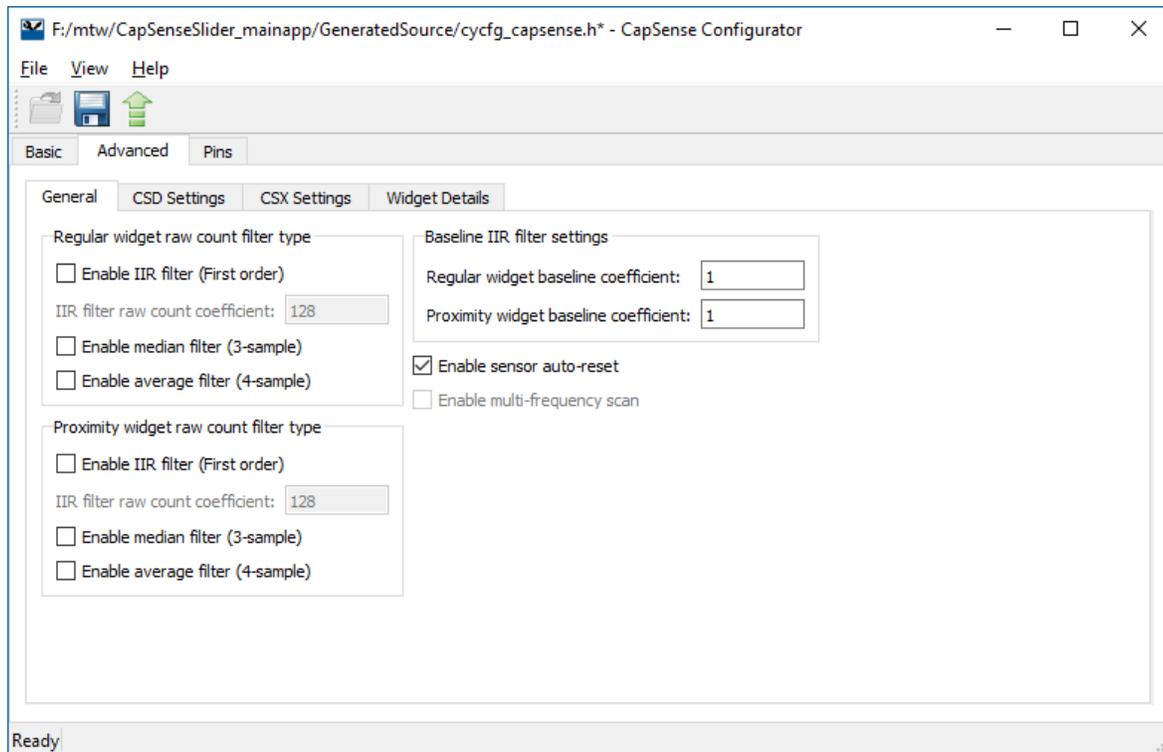
The **Advanced** tab provides advanced configuration parameters. In *SmartSense Auto-tuning*, most of the advanced parameters are automatically tuned by the algorithm and the user does not need to set values for these parameters by the *Manual* tuning process. When Manual tuning mode is selected, the **Advanced** tab allows the user to control and configure the CapSense middleware parameters.

The parameters in the **Advanced** tab are systematically arranged in the five sub-tabs.

- **General** – Contains the parameters common for all widgets respective of the sensing method used for the widgets.
- **CSD Settings** – Contains the parameters common for all widgets using the CSD sensing method. This tab is relevant only if one or more widget use the CSD sensing method.
- **CSX Settings** – Contains the parameters common for all widgets using the CSX sensing method. This tab is relevant only if one or more widget use the CSX sensing method.
- **Widget Details** – Contains parameters specific to widgets and/or sensors.

General Sub-tab

Contains the parameters common for all widgets respective of *Sensing mode* used for widgets.



The **General** sub-tab contains the following sections:

Regular widget raw count filter type

The Regular widget raw count filter type applies to raw counts of sensors belonging to non-proximity widgets. These parameters can be enabled only when one or more non-proximity widgets are added to the **Basic** tab. The filter algorithm is executed when any processing function is called by the application layer. When enabled, each filter consumes RAM to store a previous raw count (filter history). If multiple filters are enabled, the total filter history correspondingly increases so that the size of the total filter history is equal to a sum of all enabled filter histories.

Name	Description
Enable IIR filter (First order)	<p>Enables the infinite-impulse response filter (See equation below) with a step response similar to an RC low-pass filter, thereby passing the low-frequency signals (finger touch responses).</p> $Output = \frac{N}{K} \times input + \frac{(K - N)}{K} \times previous\ Output$ <p>where: <i>K</i> is always 256. <i>N</i> is the IIR filter raw count coefficient selectable from 1 to 128 in the configurator. A lower <i>N</i> (set in the <i>IIR filter raw count coefficient</i> parameter) results in lower noise, but slows down the response. This filter eliminates high-frequency noise. Consumes 2 bytes of RAM per each sensor to store a previous raw count (filter history).</p>
IIR filter raw count coefficient	<p>The coefficient (<i>N</i>) of IIR filter for raw counts is explained in the <i>Enable IIR filter (First order)</i> parameter. The range of valid values: 1-128.</p>
Enable median filter (3-sample)	<p>Enables a non-linear filter that takes three of most recent samples and computes the median value. This filter eliminates spike noise typically caused by motors and switching power supplies. Consumes 4 bytes of RAM per each sensor to store a previous raw count (filter history).</p>
Enable average filter (4-sample)	<p>The finite-impulse response filter (no feedback) with equally weighted coefficients. It takes four of most recent samples and computes their average. Eliminates periodic noise (e.g. noise from AC mains). Consumes 6 bytes of RAM per each sensor to store a previous raw count (filter history).</p>

Note If multiple filters are enabled, the execution order is the following:

- Median filter
- IIR filter
- Average filter

Proximity widget raw count filter type

The proximity widget raw count filter applies to raw counts of sensors belonging to the proximity widgets, these parameters can be enabled only when one or more proximity widgets are added on the *Basic Tab*.

Parameter Name	Description
Enable IIR filter (First order)	The design of these parameters is the same as the <i>Regular widget raw count filter</i> type parameters. The <i>Proximity</i> sensors require high-noise
IIR filter raw count coefficient	

Parameter Name	Description
Enable median filter (3-sample)	reduction. These dedicated parameters allow for setting the proximity filter configuration and behavior differently compared to other widgets.
Enable average filter (4-sample)	

Baseline filter settings

Baseline filter settings are applied to all sensors baselines. But, filter coefficients for the proximity and regular widgets can be controlled independently from each other.

The design baseline IIR filter is the same as the raw count *Enable IIR filter (First order)* parameter. But, filter coefficients can be separate for both baseline filter and raw count filters to produce a different roll-off. The baseline filter is applied to a filtered raw count (if the widget raw count filters are enabled).

Name	Description
Regular widget baseline coefficient	Baseline IIR filter coefficient selection for sensors in non-proximity widgets. The range of valid values: 1-255.
Proximity widget baseline coefficient	The design of these parameters is the same as the <i>Regular widget baseline</i> coefficient, but with a dedicated parameter allows controlling the baseline update-rate of the proximity sensors differently compared to other widgets.

General settings

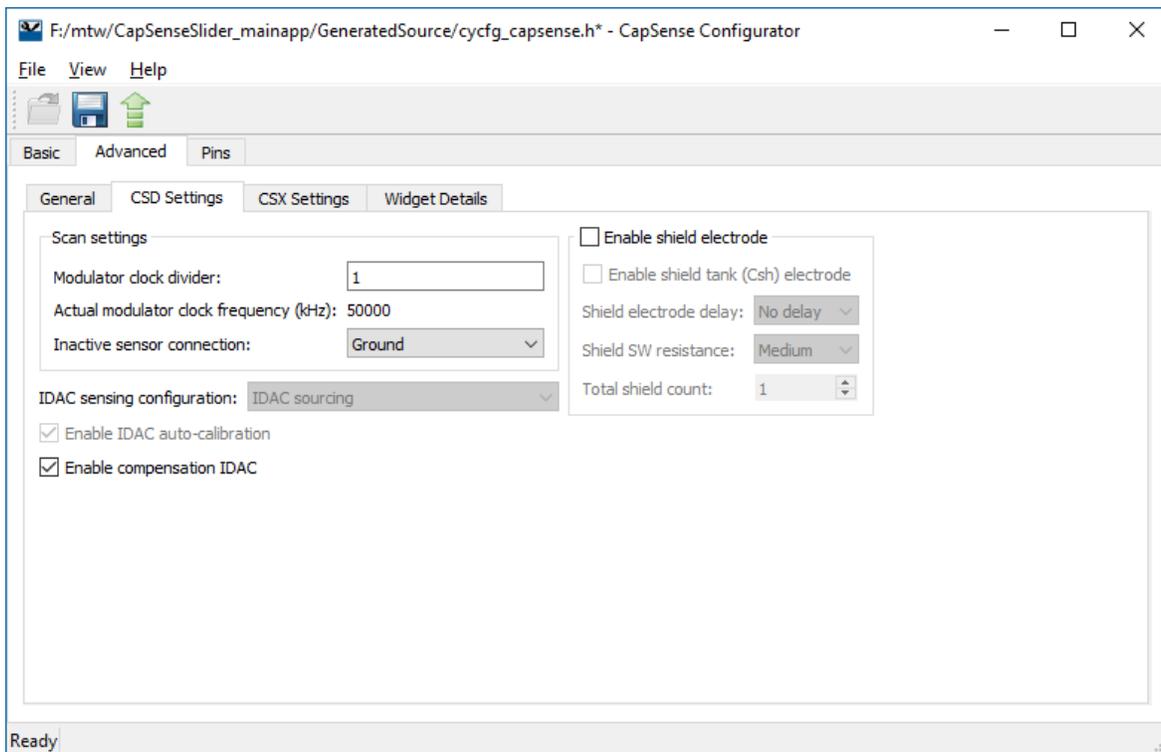
The general settings are applicable to the whole CapSense middleware behavior.

Name	Description
Enable sensor auto-reset	<p>When enabled, the baseline is always updated and when disabled, the baseline is updated only when the difference between the baseline and raw count is less than the noise threshold.</p> <p>When enabled, the feature prevents the sensors from permanently turning on when the raw count accidentally rises due to a large power supply voltage fluctuation or other spurious conditions.</p>

Name	Description
Enable multi-frequency scan	<p>The Multi-frequency scan (MFS) provides superior immunity against external noises and is suitable for applications subjected to harsh environments.</p> <p>When the Multi-frequency scan is enabled, each sensor is scanned three times with three different sensor frequencies. The base frequency F0 (zero channel) is the nominal sensor frequency. The second F1 and the third F2 frequencies are obtained by increasing the sense clock-divider by 1 and by 2 correspondingly:</p> $F_N = \frac{ModClk}{\frac{ModClk}{SnsClk} + N}$ <p>where:</p> <p><i>N</i> is a frequency channel 0, 1, or 2;</p> <p><i>F_N</i> is a scanning frequency;</p> <p><i>ModClk</i> is the CSD Modulator clock frequency Modulator clock or CSX Modulator clock frequency;</p> <p><i>SnsClk</i> is the CSD Sense clock frequency or CSX Tx clock frequency.</p> <p>The <i>SmartSense (Full Auto-Tune)</i> and the Multi-frequency scan features are mutually exclusive. If the SmartSense (Full Auto-Tune) is enabled, Multi-frequency scan cannot be enabled.</p> <p>Note</p> <ul style="list-style-type: none"> Enabling the MFS increases RAM usage by three times approximately. <p>Enabling the MFS increases the sensor scan duration by three times.</p>

CSD Settings Sub-tab

Contains the parameters common for all widgets using the *CSD sensing method*, is relevant only if at least one widget uses the CSD sensing method.



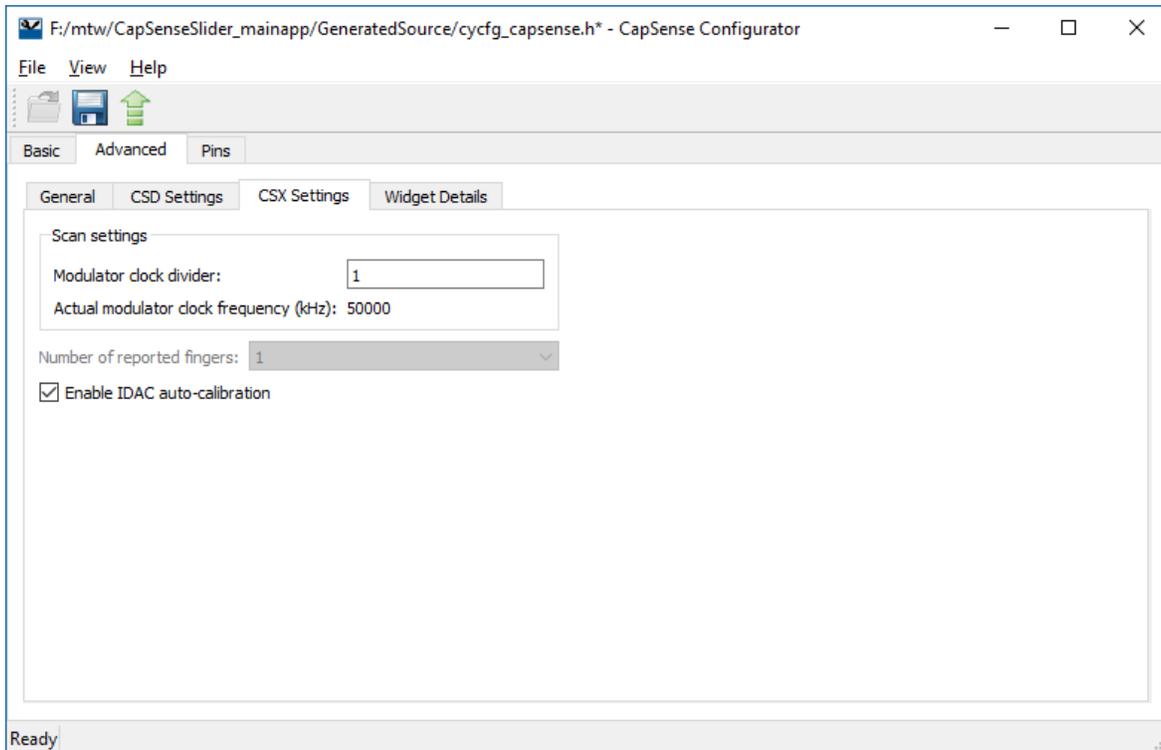
The **CSD Settings** sub-tab contains the following parameters:

Name	Description
Modulator clock divider	Selects the modulator clock divider used for the <i>CSD sensing method</i> . It defines the operating frequency of the CSD block.
Actual modulator clock frequency (kHz)	The modulator clock frequency depends on the CSD peripheral clock frequency and the modulator clock divider. The read-only value is displayed only when the CapSense Configurator is launched from the Device Configurator.
Inactive sensor connection	Selects the state of the sensor when it is not scanned. <ul style="list-style-type: none"> ▪ Ground (default) – Inactive sensors are connected to the ground. ▪ High-Z – Inactive sensors are floating (not connected to GND or Shield). ▪ Shield - Inactive sensors are connected to Shield. This option is available only if the <i>Enable shield electrode</i> check box is set. Ground is the recommended selection for this parameter when water tolerance is not required for the design. Select Shield when the design needs water tolerance or to reduce the sensor parasitic capacitance in the design.
IDAC sensing configuration	Selects the type of IDAC switching: <ul style="list-style-type: none"> ▪ IDAC Sourcing (default) – Sources current into the modulator capacitor (Cmod). The analog switches are configured to alternate between the Cmod and GND. IDAC Sourcing is recommended for most designs because of the better signal-to-noise ratio ▪ IDAC sinking – Sinks current from the modulator capacitor (Cmod). The analog switches are configured to alternate between V_{DD} and Cmod.

Name	Description
Enable IDAC auto-calibration	When enabled, values of the CSD widget IDACs are automatically set by the middleware. Select the Enable IDAC Auto-calibration parameter for robust operation. The <i>SmartSense Auto-tuning</i> parameter can be enabled only when the Enable IDAC auto-calibration is selected.
Enable compensation IDAC	The compensation IDAC is used to compensate for sensor parasitic capacitance to improve performance. Enabling the compensation IDAC is recommended unless one IDAC is required for general purpose (other than CapSense) in the application.
Enable shield electrode	The shield electrode is used to reduce the sensor parasitic capacitance, enable water-tolerant CapSense designs and enhance the detection range for the <i>Proximity</i> sensors. When the shield electrode is disabled, configurable parameters associated with the shield electrode are hidden.
Enable shield tank (Csh) capacitor	The shield tank capacitor is used to increase the drive capacity of the shield electrode driver. It should be enabled when the shield electrode capacitance is higher than 100 pF. The recommended value for a shield tank capacitor is 10nF/5V/X7R or an NP0 capacitor. The shield tank capacitor is not supported in configuration which includes both CSD and CSX sensing-based widgets.
Shield electrode delay	Configures the delay between the sensor signal and shield electrode signal for compensation of the delay added by signal routing. The four options: <ul style="list-style-type: none"> ▪ No Delay (default) ▪ 5 ns ▪ 10 ns ▪ 20 ns No Delay is the recommended value as most designs work with that option.
Shield SW resistance	Selects the resistance of switches used to drive the shield electrode. The four options: <ul style="list-style-type: none"> ▪ Low ▪ Medium (default) ▪ High ▪ Low EMI
Total shield count	Selects the number of shield electrodes required in the design. Most designs work with one dedicated shield electrode but, some designs require multiple dedicated shield electrodes to ease the PCB layout routing or to minimize the PCB area used for the shield layer. The minimum value is 0 (i.e. shield signal could be routed to sensors using the <i>Inactive sensor connection</i> parameter) and the maximum value is equal to the total number of CapSense-enabled port pins available for the selected device.

CSX Settings Sub-tab

The parameters in this sub-tab apply to all widgets that use the *CSX sensing method*, is relevant only if at least one widget uses the CSX sensing method.

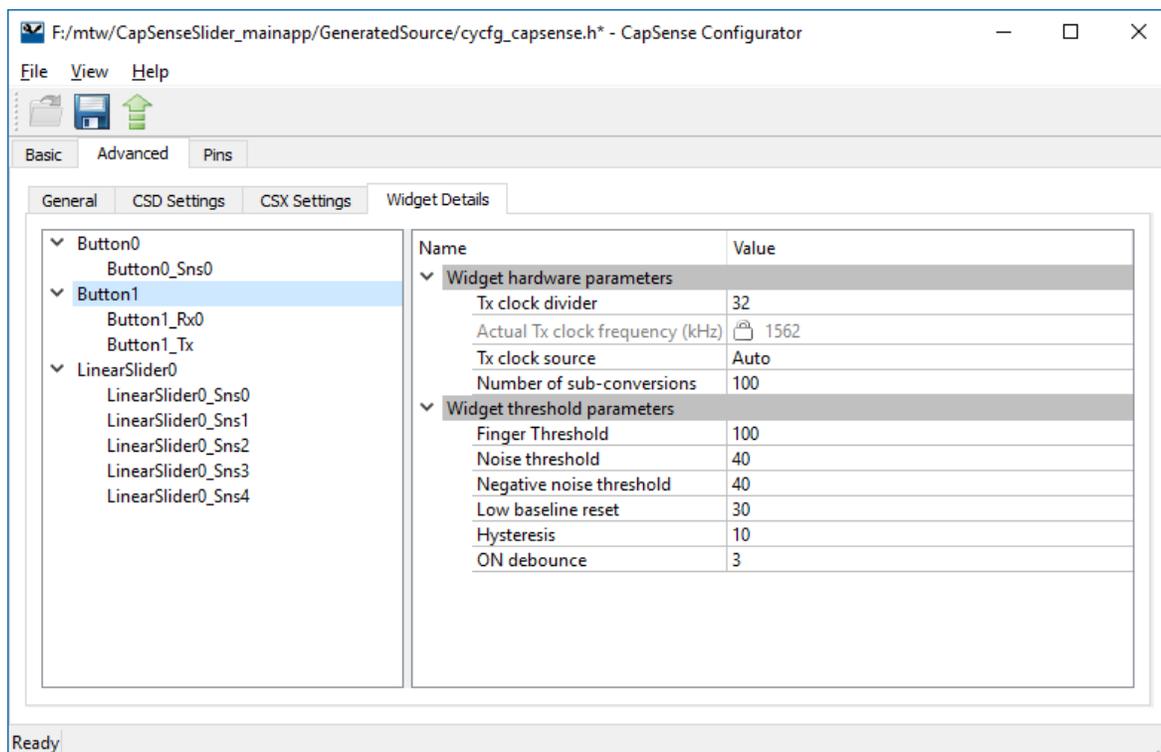


The **CSX Settings** sub-tab contains the following parameters:

Name	Description
Modulator clock divider	Selects the modulator clock divider used for the <i>CSX sensing method</i> . It defines the operating frequency of the CSD block. A higher modulator clock frequency reduces the sensor scan time, results in lower power, and reduces the noise in raw counts, so use the highest possible frequency.
Actual modulator clock frequency (kHz)	The modulator clock frequency depends on the CSX peripheral clock frequency and the modulator clock divider. The read-only value is displayed only when the CapSense Configurator is launched from the Device Configurator.
Number of reported fingers	Sets the number of reported fingers for a CSX Touchpad widget only. The available options are from 1 to 3.
Enable IDAC auto-calibration	When enabled, IDAC values are automatically set by the middleware. It is recommended to select the Enable IDAC auto-calibration for robust operation.

Widget Details Sub-tab

This sub-tab contains parameters specific to each widget and sensor. These parameters must be set when *SmartSense Auto-tuning* is not enabled. The parameters are unique for each widget type.



The **Widget Details** sub-tab contains the following parameters:

Name	Description
Widget General Parameters	
Diplexing	Enabling Diplexing allows doubling the slider physical touch sensing area by using the specific diplexing sensor pattern and without using additional port pins and sensors.
Maximum position	Represents the maximum Centroid position for the slider. A touch on the slider would produce a position value from 0 to the maximum position-value set. No Touch would produce 0x0000.
Maximum X-axis position	Represents the maximum column (X-axis) Centroid position and row (Y-axis) Centroid positions for a touchpad. A touch on the touchpad would produce a position value from 0 to the maximum position set. No Touch would produce 0x0000.
Maximum Y-axis position	
Widget Hardware Parameters	
Note All the Widget Hardware parameters for the CSD widgets are automatically set when <i>SmartSense Auto-tuning</i> is selected in the <i>CSD tuning mode</i> .	
Sense clock divider	Sets the CSD Sense clock divider. When SmartSense is selected in <i>CSD tuning mode</i> , the Sense Clock divider is automatically set by the middleware to an optimal value by following the 2*5*R*C rule (refer to CapSense design guide for more information on this rule) and this control is not editable.

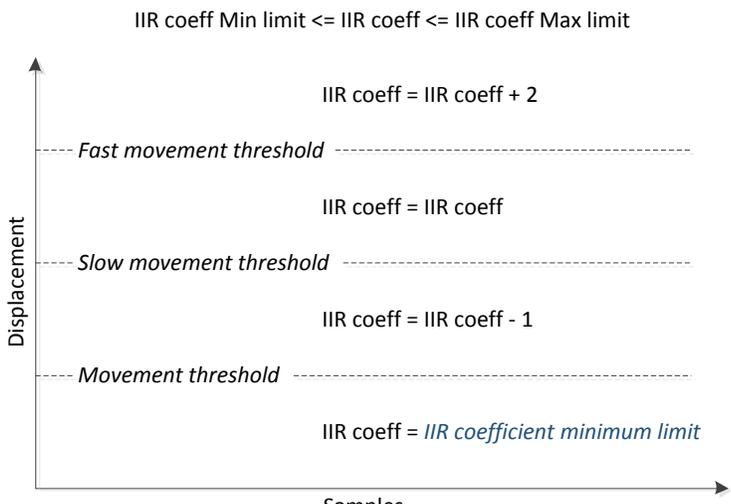
Name	Description
Row sense clock divider	Sets the CSD Sense clock divider for row and column sensors of the <i>Matrix Buttons</i> and <i>Touchpad</i> widgets.
Column sense clock divider	
Tx clock divider	Sets the Tx Clock divider for the CSX widgets.

Name	Description
Sense clock source	<p>The Sense clock frequency is derived from the Modulator clock frequency using a clock-divider and is used to sample the sensor. Both the clock source and clock divider are configurable.</p> <p>The Spread Spectrum Clock (SSC) provides a dithering clock source with a center frequency equal to the Sense clock frequency. The PRS clock source spreads the clock using the pseudo-random sequencer and the Direct source disables both SSC and PRS sources and uses a fixed-frequency clock.</p> <p>Both PRS and SSC reduce the radiated noise by spreading the clock and improve the immunity against external noise. Using a higher number of bits of SSC and PRS lowers the radiation and increases the immunity against external noise.</p> <p>The following sources are available:</p> <p><i>Direct</i> – PRS and SSC are disabled and a fixed clock is used.</p> <p><i>PRS8</i> – The clock spreads using PRS to Modulator Clock / 256.</p> <p><i>PRS12</i> – The clock spreads using PRS to Modulator Clock / 4096.</p> <p><i>SSC6, SSC7, SSC9 and SSC10</i> – The clock spreads using from 6 to 10 bits of the sense-clock divider respectively.</p> <p><i>Auto</i> – The middleware automatically selects optimal SSC, PRS or Direct sources individually for each widget. The Auto is the recommended sense clock source selection.</p> <p>The following rules and recommendations for the SSC selection:</p> <p>The ratio between the Modulator clock frequency $ModClk$ and Sense clock frequency must be greater than or equal to 20.</p> <p>20% of the ratio between the Modulator clock frequency $ModClk$ and Sense clock frequency should be greater or equal to the SSC frequency range = 32. It allows varying the ratio between the Modulator and Sense clock frequencies to 32 different clocks evenly spaced over +/- 10% from the center frequency.</p> $160 \leq \frac{ModClk}{SnsClk}$ <p>Where $ModClk$ is the Modulator clock frequency and $SnsClk$ is Sense clock frequency.</p> <p>It is recommended that at least one full-spread spectrum polynomial should end during the scan time:</p> $\frac{2^N - 1}{ModClk} \geq \frac{2^{SSCN} - 1}{SnsClk}$ <p>where N is the <i>Scan resolution</i>, $SSCN$ is the number of bits used for SSC (6, 7, 9 and 10), $ModClk$ is Modulator clock frequency and $SnsClk$ is Sense clock frequency.</p> <p>It is recommended that the number of sub-conversions for the widget should be an integer multiple of the SSC polynomial selected. For example, if SSC6 is selected, the number of the sub-conversion should be multiple of $(2^{SSC6}-1) = 63$.</p> <p>The recommendation for the PRS selection:</p> <p>At least one full PRS polynomial should finish during the scan time:</p> $\frac{2^N - 1}{ModClk} \geq \frac{2^{PRSN} - 1}{SnsClk}$ <p>where N is the <i>Scan resolution</i>, $PRSN$ is the number of bits used for PRS (8 and 12), $ModClk$ is the Modulator clock frequency $ModClk$ and $SnsClk$ is the average Sense clock frequency.</p>

Name	Description
Tx clock source	<p>The Tx clock frequency derives from the Modulator clock frequency using a clock-divider and is used to sample the sensor. Both the clock source and clock divider are configurable.</p> <p>The Spread Spectrum Clock (SSC) provides a dithering clock source with a center frequency equal to the Tx clock frequency and the Direct source disables the SSC source and uses a fixed frequency clock. The SSC reduces the radiated noise by spreading the clock and improves the immunity against external noise. Using a higher number of bits of SSC lowers the radiation and increases the immunity against external noise.</p> <p>The following clock sources are available:</p> <p><i>Direct</i> – SSC is disabled and a fixed clock is used.</p> <p><i>SSC6, SSC7, SSC9 and SSC10</i> – The clock spreads using from 6 to 10 bits of the sense-clock divider respectively.</p> <p><i>Auto</i> – The middleware automatically selects optimal SSC or Direct sources individually for each widget. Auto is the recommended Sense clock source selection.</p> <p>The rules and recommendations for the SSC selection:</p> <p>The ratio between the Modulator clock frequency and Tx clock frequency must be greater than or equal to 20.</p> <p>20% of the ratio between the Modulator clock frequency and Tx clock frequency should be greater or equal to the SSC frequency range = 32. It allows varying the ratio between the Modulator and Tx clock frequencies to 32 different clocks evenly spaced over +/- 10% from the center frequency.</p> $160 \leq \frac{ModClk}{TxClk}$ <p>where <i>ModClk</i> is the Modulator clock frequency and <i>TxClock</i> is Tx clock frequency.</p> <p>It is recommended that at least one full-spread spectrum polynomial should end during the scan time.</p> $\frac{N_{Sub}}{ModClk} \geq \frac{2^{SSCN} - 1}{TxClk}$ <p>where <i>N_{Sub}</i> is the <i>Number of sub-conversions</i>, <i>SSCN</i> is the number of bits used for SSC (6, 7, 9 and 10), <i>ModClk</i> is the Modulator clock frequency and <i>TxClock</i> is the Tx clock frequency.</p> <p>It is recommended that <i>Number of sub-conversions</i> for the widget should be an integer multiple of the SSC polynomial selected. For example, if SSC6 is selected, the number of sub-conversion should be multiple of $(2^{SSC6}-1) = 63$.</p>
Scan resolution	Selects the scan resolution of the CSD widgets (Resolution of capacitance to digital conversion). Acceptable values are from 6 to 16 bits.
Number of sub-conversions	Selects the number of sub-conversions in the CSX sensing method .
Modulator IDAC	Sets the modulator IDAC value for the CSD Button, Slider, or Proximity widget. The value of this parameter is automatically set when Enable IDAC auto-calibration is selected in the CSD Settings tab.
Row modulator IDAC	Sets a separate modulator IDAC value for the row and column sensors of the CSD Matrix Buttons and Touchpad widget.
Column modulator IDAC	These parameters values are automatically set when Enable IDAC auto-calibration is checked in the CSD Settings tab.

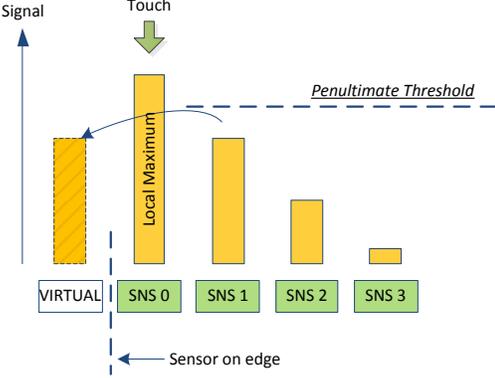
Name	Description
Widget Threshold Parameters	
Note All the threshold parameters for the CSD widgets are automatically set when <i>SmartSense (Full Auto-Tune)</i> is selected in the <i>CSD tuning mode</i> parameter.	
Finger threshold	<p>The finger threshold parameter is used along with the hysteresis parameter to determine the sensor state as follows:</p> <ul style="list-style-type: none"> ▪ ON – $\text{Signal} > (\text{Finger Threshold} + \text{Hysteresis})$ ▪ OFF – $\text{Signal} \leq (\text{Finger Threshold} - \text{Hysteresis})$. <p>Note that “Signal” in the above equations refers to: Difference Count = Raw Count – Baseline.</p> <p>It is recommended to set the Finger threshold parameter value equal to the 80% of the touch signal.</p> <p>The Finger Threshold parameter is not available for the <i>Proximity</i> widget. Instead, Proximity has two thresholds:</p> <ul style="list-style-type: none"> ▪ <i>Proximity threshold</i> ▪ <i>Touch threshold</i>
Noise threshold	<p>Sets a raw count limit below which a raw count is considered as noise. When a raw count is above the Noise Threshold, a difference count is produced and the baseline is updated only if <i>Enable sensor auto-reset</i> is selected (In other words, the baseline remains constant as long as the raw count is above the baseline + noise threshold. This prevents the baseline from following raw counts during a finger touch detection event).</p> <p>It is recommended to set the noise threshold parameter value equal to 2x noise in the raw count or the 40% of the signal.</p>
Negative noise threshold	<p>Sets a raw count limit below which the baseline is not updated for the number of samples specified by the <i>Low baseline reset</i> parameter.</p> <p>The negative noise threshold ensures that the baseline does not fall low because of any high-amplitude repeated negative-noise spikes on a raw count caused by different noise sources such as ESD events.</p> <p>It is recommended to set the negative noise threshold parameter value equal to the <i>Noise threshold</i> parameter value.</p>
Low baseline reset	<p>This parameter is used along with the <i>Negative noise threshold</i> parameter. It counts the number of abnormally low raw counts required to reset the baseline.</p> <p>If a finger is placed on the sensor during a device startup, the baseline gets initialized to a high raw count value at a startup. When the finger is removed, the raw count falls to a lower value. In this case, the baseline should track low raw counts. The Low Baseline Reset parameter helps handle this event. It resets the baseline to a low raw count value when the number of low samples reaches the low-baseline reset number.</p> <p>Note After a finger is removed from the sensor, the sensor will not respond to finger touches for low baseline-reset time.</p> <p>The recommended value is 30 which works for most designs.</p>
Hysteresis	<p>The hysteresis parameter is used along with the <i>Finger threshold</i> parameter (<i>Proximity threshold</i> and <i>Touch threshold</i> for Proximity sensor) to determine the sensor state. The hysteresis provides immunity against noisy transitions of the sensor state.</p> <p>See the description of the <i>Finger threshold</i> parameter for details.</p> <p>The recommend value for the hysteresis is the 10% <i>Finger threshold</i>.</p>

Name	Description
ON debounce	<p>Selects a number of consecutive CapSense scans during which a sensor must be active to generate an ON state from the middleware. The Debounce ensures that high-frequency, high-amplitude noise does not cause false detection</p> <ul style="list-style-type: none"> ▪ Buttons/Matrix buttons/Proximity – An ON status is reported only when the sensor is touched for a consecutive debounce number of samples. ▪ Sliders/Touchpads – The position status is reported only when any of the sensors is touched for a consecutive debounce number of samples. <p>The recommended value for the Debounce parameter is 3 for reliable sensor status detection.</p>
Proximity threshold	<p>The design of these parameters is the same as for the <i>Finger threshold</i> parameters. The proximity sensor requires a higher noise reduction, and supports two levels of detection:</p>
Touch threshold	<ul style="list-style-type: none"> ▪ The proximity level to detect an approaching hand or finger. ▪ The touch level to detect a finger touch on the sensor similarly to other <i>Widget Type</i> sensors. <p>Note that for valid operation, the Proximity threshold must be higher than the Touch threshold.</p> <p>The threshold parameters such as <i>Hysteresis</i> and <i>ON</i> debounce are applicable to both detection levels.</p>
Velocity	<p>Defines the maximum speed of a finger movement in terms of the squared distance of the touchpad resolution. The parameter is applicable for a multi-touch touchpad (CSX Touchpad) only. If the detected position of the next scan is further than the defined squared distance, then this touch is considered as a separate touch with a new touch ID.</p>
<p>Position Filter Parameters</p>	
<p>These parameters enable firmware filters on a centroid position to reduce noise. These filters are available for Slider and Touchpad widgets only. If multiple filters are enabled, the execution order corresponds to the listed below and the total RAM consumption increases so that the size of the total filter history is equal to a sum of all enabled filter histories.</p>	
IIR filter	<p>Enables the infinite-impulse response filter (see equation below) with a step response.</p> $Output = \frac{N}{K} \times Input + \frac{(K - N)}{K} \times prev\ Output$ <p>where: <i>K</i> is always 256; <i>N</i> is the IIR filter raw count coefficient selectable from 1 to 255 in the configurator. A lower <i>N</i> (set in the <i>IIR filter coefficient</i> parameter) results in lower noise, but slows down the response. This filter eliminates high-frequency noise. Consumes 2 bytes of RAM per each position (filter history).</p>
IIR filter coefficient	<p>The coefficient (<i>N</i>) of the IIR filter for a position as explained in the <i>IIR filter</i> parameter. The range of valid values: 1-255.</p>
Median filter	<p>Enables a non-linear filter that takes three of most recent samples and computes the median value. This filter eliminates the spikes noise typically caused by motors and switching power supplies. Consumes 4 bytes of RAM per each position (filter history).</p>
Average filter	<p>Enables the finite-impulse response filter (no feedback) with equally weighted coefficients. It takes two of most recent samples and computes their average. Eliminates periodic noise (e.g. noise from AC mains). Consumes 2 bytes of RAM per each position (filter history).</p>

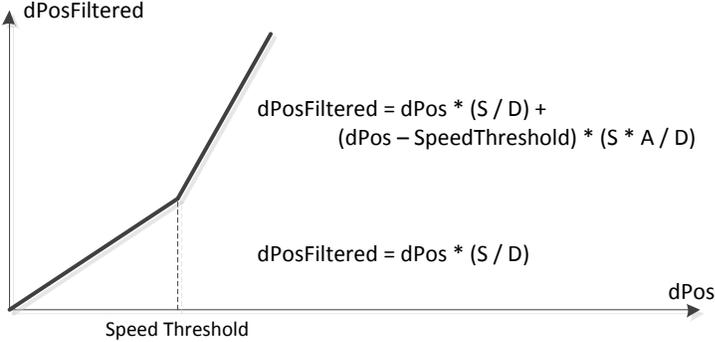
Name	Description
Jitter filter	This filter eliminates the noise in the position data that toggles between the two most recent values. If the most recent position value is greater than the previous one, the current position is decremented by 1; if it is less, the current position is incremented by 1. The filter is most effective at low noise. Consumes 2 bytes of RAM per each position (filter history).
<p>Adaptive IIR Filter Parameters</p> <p style="text-align: center;">IIR coeff Min limit <= IIR coeff <= IIR coeff Max limit</p> 	
Adaptive IIR filter	Enables the Adaptive IIR filter. It is the IIR filter that changes its IIR coefficient according to the speed of the finger movement. This is done to smooth the fast movement of the finger and at the same time control properly the position movement. The filter coefficients are automatically adjusted by the adaptive algorithm with the speed of the finger movement. If the finger moves slowly, the IIR coefficient decreases; if the finger moves fast, the IIR coefficient increases from the existing value. Consumes 3 bytes of RAM per each position (filter history).
Position movement threshold	Defines the position threshold below which a position displacement is ignored or considered as no movement.
Position slow movement threshold	Defines the position threshold below which (and above Position movement threshold) a position displacement (the difference between the current and previous position) is considered as slow movement. If the position displacement is within the threshold limits, the IIR filter coefficient decreases during each new scan. So, the filter impact on the position becomes less intensive.
Position fast movement threshold	Defines the position threshold above which a position displacement is considered as fast movement. If the position displacement is above the threshold limit, the IIR filter impact on the position becomes more intensive during each new scan as the filter coefficient increases.
IIR coefficient maximum limit	Defines the maximum limit of the IIR coefficient when the finger moves fast. The fast movement event is defined by the Position fast movement threshold.
IIR coefficient minimum limit	Defines the minimum limit of the IIR coefficient when the finger moves slowly. The slow movement event is defined by the Position slow movement threshold.

Name	Description
IIR coefficient divisor	<p>This parameter acts as the scale factor for the filter IIR coefficient.</p> $Output = \frac{Coeff}{Divisor} \times Input + \frac{Divisor - Coeff}{Divisor} \times previous Output$ <p>where: <i>Input</i>, <i>Output</i>, and <i>Previous Output</i> are the touch positions; <i>Coeff</i> is the automatically adjusted IIR filter coefficient; <i>Divisor</i> is the IIR coefficient divisor (this parameter).</p>
<p>Touchpad Parameters These parameters are available for the CSD Touchpad widgets only.</p>	
Centroid type	<p>Selects a sensor matrix size for centroid calculation. The 5x5 centroid (also known as Advanced Centroid) provides benefits such as Two finger detection, Edge correction and improved accuracy.</p> <p>If Advanced Centroid is selected, the below parameters are configured as well.</p>
Cross-coupling position threshold	<p>Defines the cross coupling threshold. This value is subtracted from the sensor signal used for centroid position calculation to improve the accuracy.</p> <p>The threshold should be equal to a sensor signal when a finger is near the sensor but is not touching the sensor. This can be determined by slowly dragging the finger across the panel and finding the inflection point of the difference counts at the base of the curve. The difference value at this point is the Cross-coupling threshold. The default value is 5.</p>
Edge correction	<p>This feature is available if the Centroid Type is configured to 5x5.</p> <p>When enabled, a matrix of centroid calculation is updated with virtual sensors on the edges of a touchpad. It improves the accuracy of the reported position on the edges. When enabled, two more parameters must be configured: Virtual Sensor threshold and Penultimate threshold.</p>

Name	Description
<p>Virtual sensor threshold</p>	<p>This parameter is applicable only if Edge correction is enabled and it is used to calculate a signal (difference count) for a virtual sensor used for the edge correction algorithm.</p> <p>A touch position on a slider or touchpad is calculated using a signal from the local-maxima sensor and its neighboring sensors. A touch on the edge sensor of a slider or touchpad does not accurately report a position because the edge sensor lacks signal from one side of neighboring sensors of the local-maxima sensor.</p> <p>If the Edge correction is enabled, the algorithm adds a virtual neighbor sensor to correct the deviation in the reported position. The Virtual sensor signal is defined by the Virtual sensor threshold:</p> $DiffCount_{VIRTUAL} = (Threshold_{VIRTUAL} - DiffCount_{SNS0}) \times 2$ <p>where:</p> <ul style="list-style-type: none"> $DiffCount_{VIRTUAL}$ is the virtual sensor difference count; $Threshold_{VIRTUAL}$ is the virtual sensor threshold; $DiffCount_{SNS0}$ is the sensor 0 difference count. <p>The conditions for a virtual sensor (and Edge correction algorithm) to be applied:</p> <ul style="list-style-type: none"> Local-maxima detected on the edge sensor <p>Difference count from the penultimate sensor less than the Penultimate threshold.</p>

Name	Description
Penultimate threshold	<p>This parameter is applicable only if the Edge correction is enabled and it works along with the Virtual sensor threshold parameter.</p> <p>This parameter defines the threshold of penultimate sensor signal. If the signal from penultimate sensor is below the Penultimate threshold, the edge correction algorithm is applied to the centroid calculation.</p> <p>The conditions for the edge correction to be applied:</p> <ul style="list-style-type: none"> ▪ Local-maxima detected on the edge sensor ▪ The difference count of the penultimate sensor (SNS 1 in the figure below) less than the Penultimate threshold. 

Name	Description
<p>Two finger detection</p>	<p>Enables the detection of the second finger on a CSD touchpad.</p> <p>In general, a CSD touchpad can detect only one true touch position. A CSD touchpad widget consists of two Linear Sliders and each slider reports the X and Y coordinates of a finger touch. If there are two touches on the touchpad, there are four possible touch positions as shown in the figure below. The two of these touches are real touches and two are known as “ghost” touches. There is no possibility to differentiate between ghost and real touches in a CSD widget (to get true multi-touch performance, use the CSX Touchpad widget).</p> <div data-bbox="613 520 1242 1010" data-label="Diagram"> </div> <p>But, if this feature is enabled, the CSD touchpad can report up to two touches, mainly to be used in conjunction with two-finger gestures where real and ghost touches do not need to be fully differentiated. It is available for the CSD touchpad only when the Centroid type is configured to 5x5.</p> <p>The Advanced centroid (Centroid type is 5x5) uses the 3x3 centroid matrix when detects two touches.</p>
<p>Acceleration coefficient</p>	<p>Defines the value at which the position movement needs to be interpolated when the movement is classified as fast movement. The reported position displacement is multiplied by this parameter.</p>
<p>Speed coefficient</p>	<p>Defines the value at which the position movement is interpolated when the movement is classified as slow movement. The reported position displacement is multiplied by this parameter.</p>
<p>Divisor value</p>	<p>Defines the divisor value used to create a fraction for the acceleration and speed coefficients. The interpolated position coordinates are divided by the value of this parameter.</p>
<p>X-axis speed threshold</p>	<p>Defines the threshold to distinguish fast and slow movement on the X axis. If the X-axis position displacement reported between two consecutive scans exceeds this threshold, then it is considered as fast movement, otherwise as slow movement.</p>
<p>Y-axis speed threshold</p>	<p>Defines the threshold to distinguish fast and slow movement on the Y axis. If the Y-axis position displacement reported between two consecutive scans exceeds this threshold, then it is considered as fast movement, otherwise as slow movement.</p>

Name	Description
Ballistic multiplier	<p>Enables the Ballistic multiplier filter used to provide better user experience of the pointer movement. Fast movement will move the cursor by more pixels. Consumes 16 bytes of RAM when enabled.</p> <p>The simplified diagram of the Ballistic Multiplier filter operation:</p>  <p>where,</p> <p>$dPos$ is an input position displacement either in the X axis or Y axis,</p> <p>$dPosFiltered$ is the filtered displacement;</p> <p>$SpeedThreshold$ is either the X-axis speed threshold or Y-axis speed threshold;</p> <p>A is the Acceleration coefficient;</p> <p>S is the Speed coefficient;</p> <p>D is the Divisor value.</p>
Gesture Parameters	
Enable gestures	<p>Master enable for gestures feature.</p> <p>Each gesture consists of a sequence of Touchdown and Lift Off events.</p> <p>A simple touch on a widget is reported as a Touchdown event.</p> <p>Removal of a finger from a widget reported as a Lift Off event. If the Lift Off event triggers another higher-level Gesture, then the Lift Off event is not reported.</p>
Enable one-finger single click gestures	<p>One-finger single click gesture is a combination of a Touchdown and Lift Off events with the conditions to be met:</p> <ul style="list-style-type: none"> ▪ A touchdown event is followed by a Lift Off event. ▪ The touch duration (duration between touchdown and lift off) must be greater than Minimum click timeout and less than Maximum click timeout. ▪ Position displacements between the Touchdown and Lift Off events must be within the Maximum click distance.
Enable one-finger double click gestures	<p>A One-finger double click gesture is a combination of two sequential one-finger single click gestures under specific conditions:</p> <ul style="list-style-type: none"> ▪ Both clicks in the sequence must meet one-finger single click conditions. ▪ The touch duration between the two touchdown events must be within the Minimum second click interval and Maximum second click interval timeout limits. ▪ The distance between two clicks must not exceed the Maximum second click distance

Name	Description
Enable one-finger click & drag gestures	<p>This gesture is a one-finger click and then a hold, followed by a drag. A typical use case is while moving items on the screen from one point to another. It is triggered when the finger movement follows this sequence: Touchdown → Lift Off → Touchdown → Drag</p> <p>Gesture triggering condition: A one-finger click gesture and a subsequent touchdown were detected within the Minimum click timeout and Maximum click timeout limits and within Maximum second click distance. Then the finger exceeds the Maximum click distance from a drag touchdown.</p>
Enable two-finger single click gestures	<p>A Two-finger single click gesture is a combination of a Touchdown and Lift Off events with under specific conditions:</p> <ul style="list-style-type: none"> ▪ Two simultaneous finger touches (touchdown and lift off) should be detected. ▪ The duration between the second finger touchdown and lift off events of both fingers must be within the Minimum second click interval and Maximum second click interval timeout limits. The duration counting starts when the settling time elapsed for the second finger touchdown event. ▪ A position displacement between the touchdown and lift off events must be less than the Maximum second click distance.
Enable one-finger scroll gestures	<p>A One-finger Scroll gesture is a combination of a touchdown followed by a displacement in a specific direction under specific conditions:</p> <ul style="list-style-type: none"> ▪ For a slider, the position displacement between two consecutive scans must exceed the Minimum scroll distance. ▪ The Scroll debounce number of a scroll gesture in the same direction is already detected.
Enable two-finger scroll gestures	<p>The design of a two-finger scroll gesture is the same as of a one-finger scroll gesture, except for the conditions below.</p> <ul style="list-style-type: none"> ▪ The conditions of a one-finger scroll are met. ▪ There must be two simultaneous finger touches detected on a widget for a scroll to be considered as a two-finger scroll. ▪ The displacement of both finger touches must be on same direction for a two-finger scroll to be valid.
Enable one-finger edge swipe gestures	<p>An edge swipe gesture is a combination of a touchdown on an edge followed by a displacement towards the center.</p> <p>The conditions for an edge swipe gesture:</p> <ul style="list-style-type: none"> ▪ A touchdown event must occur in the edge area defined by the Edge size. ▪ A finger displacement must occur from the edge towards the center within the Maximum edge angle. ▪ The displacement must exceed the Minimum edge distance within the Maximum edge timeout duration.

Name	Description
Enable one-finger flick gestures	<p>A flick gesture is a combination of a touchdown followed by a high-speed displacement and a lift off event.</p> <p>A flick gesture starts at a touchdown and ends and reported at a lift off event. The conditions for a flick gesture.</p> <ul style="list-style-type: none"> ▪ The displacement must exceed the Minimum flick distance. ▪ The duration between a touchdown and lift off events must be less than the Maximum flick timeout. <p>Note The flick gesture is detected in 8 directions:</p> <ul style="list-style-type: none"> ▪ Up ▪ Down ▪ Left ▪ Right ▪ Up-Right ▪ Down-Left ▪ Up-Left ▪ Down-Right
Enable one-finger rotate gestures	<p>A one-finger rotate gesture is reported when a circular displacement is detected. The decoding algorithm uses four directions to identify a circular displacement. A displacement in all four directions must be in the succession order to report a rotate gesture. The rotation direction can be clockwise or counter-clockwise.</p>
Enable two-finger zoom gestures	<p>A two-finger zoom gesture is reported when two touches move towards each other (Zoom Out) or move away from each other (Zoom In).</p> <p>The conditions for a zoom gesture:</p> <ul style="list-style-type: none"> ▪ An increase or decrease in distance between two-finger touch positions must exceed the Minimum zoom distance. ▪ The zoom debounce number of a Zoom In or Zoom Out gesture must be sequentially detected for a Zoom gesture to be reported. <p>A scroll to the zoom debounce number of a zoom gestures must be sequentially detected for a Zoom gesture to be reported. If a Zoom gesture occurred after a scroll, the gesture is reported and there was no lift off event between the scroll and Zoom gestures.</p>

Name	Description
Enable gesture filtering	Enables filtering of the detected gestures. The gesture priority defined as follow (starting from the most important): <ul style="list-style-type: none"> • Two-finger zoom • Two-finger scroll • One-finger rotate • One-finger edge swipe • One-finger flick • One-finger scroll • Two-finger single click • One-finger click and drag • One-finger double click • One-finger single click • Touchdown • Liftoff
Maximum click timeout	Defines the maximum duration between a touchdown and lift off events of a click event. This parameter is used in all click-based gestures.
Minimum click timeout	Defines the minimum duration between a touchdown and lift off events of a click event. This parameter is used in all click-based gestures.
Maximum click distance	Defines the maximum displacement between a touchdown and lift off events of a click event. This parameter is used in all click-based gestures.
Maximum second click interval	Defines the maximum displacement between a touchdown and lift off events of a click event. This parameter is used in all click-based gestures.
Minimum second click interval	This parameter defines the minimum duration between the first lift off and the second touchdown events. If the second click occurs early this limit, the double click and click&drag gestures are not reported.
Maximum second click distance	Defines the maximum distance between the first lift off event and the second touchdown event. If the second click occurs outside this limit, the double click and click&drag gestures are not reported.
Scroll debounce	Defines the minimum number sequential scroll steps in the same direction to be detected prior to the scroll is considered valid. A widget must detect scroll steps, at the minimum of Debounce times in the same direction to be considered as a scroll in that direction.
Minimum scroll distance	Defines the minimum displacement to recognize a single scroll step. A scroll step is calculated between two consecutive scans.
Rotate debounce	Defines a maximum number of sequential rotate steps in the same direction to deem a rotate gesture invalid. For example, if the Debounce value is set to 5, then the touch cannot continue in the same direction for 5 rotate steps and still have a valid rotate gesture. After this threshold, the reported gesture stops being a rotate gesture. If this parameter is set to 0, then the Debounce is disabled.
Minimum rotate distance	Defines the minimum displacement to recognize a single rotate step.
Zoom debounce	Defines the minimum number of zoom steps in a particular direction (in or out) to report a zoom gesture.

Name	Description
Minimum zoom distance	Defines the minimum displacement to recognize a single zoom step.
Maximum flick timeout	Defines the maximum duration of how long a flick gesture is searched after a touchdown event. A position displacement and lift off event must happen within the duration defined by this parameter for a flick to be valid.
Minimum flick distance	Defines the minimum displacement to be detected for a one-finger flick to be valid.
Edge size	Defines the maximum edge area where a touchdown must be detected for an edge swipe to be reported.
Minimum edge distance	Defines the minimum displacement to be detected from an edge to the center for an edge swipe to be reported.
Maximum edge timeout	Defines the maximum duration within which an edge swipe must occur to be reported. The displacement must exceed the displacement threshold within the duration defined by this parameter for the edge swipe to be reported.
Maximum edge angle	To report this gesture a finger movement should start from an edge and moves in the center direction. This is an ideal line. This parameters defines the maximum angle deviation (in degree) from this ideal line for edge swipe to be valid. Degree 1 means that the user can do gestures only on a single ideal line.
Sensor Parameters	
Compensation IDAC value	<p>Sets the Compensation IDAC value for each CSD sensor when <i>Enable compensation IDAC</i> is selected on the <i>CSD Settings</i> tab. If the <i>CSD tuning mode</i> is set to <i>SmartSense Auto-tuning</i> or <i>Enable IDAC auto-calibration</i> is selected on the <i>CSD Settings</i> tab, the value of this parameter is set equal to the Modulator IDAC value at a device power-up for the maximum performance from the sensor.</p> <p>Select the <i>Enable IDAC auto-calibration</i> for robust operation.</p>
IDAC Values	<p>Sets the IDAC value for each CSX sensor/node, a lower IDAC value without saturating raw counts provides better performance for sensor/nodes.</p> <p>When <i>Enable IDAC auto-calibration</i> is selected on the <i>CSX Settings</i> tab, the value of this parameter is automatically set to the lowest possible value at a device power-up for better performance.</p> <p>It is recommended to select <i>Enable IDAC auto-calibration</i> for robust operation.</p>
Selected pins	Selects a port pin for the sensor (CSD sensing) and electrode (CSX sensing). The available options use a dedicated pin for a sensor or re-use one or more pins from any other sensor. Re-using the pins of any other sensor from any widgets helps create a ganged sensor.

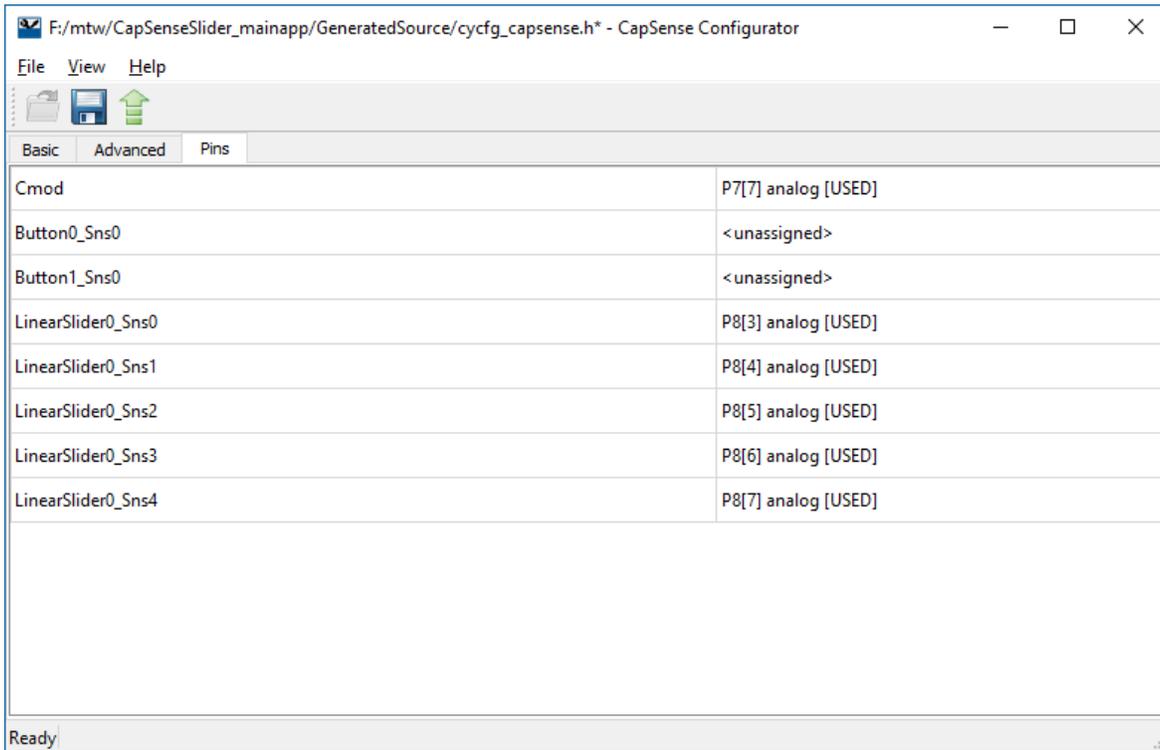
The following table shows which Widget / Sensor parameters belong to a given widget type:

Parameters	Widget Type								
	CSD Widget						CSX Widget		
	Button	Linear Slider	Radial Slider	Matrix Buttons	Touchpad	Proximity	Button	Matrix Buttons	Touchpad
Widget General									
Diplexing		√							
Maximum position		√	√						
Maximum X-axis position					√				√
Maximum Y-axis position					√				√

Parameters	Widget Type								
	CSD Widget						CSX Widget		
	Button	Linear Slider	Radial Slider	Matrix Buttons	Touchpad	Proximity	Button	Matrix Buttons	Touchpad
Widget Hardware									
Sense clock divider	√	√	√			√			
Row sense clock divider				√	√				
Column sense clock divider				√	√				
Sense clock source	√	√	√	√	√	√			
Tx clock divider							√	√	√
Tx clock source							√	√	√
Scan resolution	√	√	√	√	√	√			
Number of sub-conversions							√	√	√
Modulator IDAC	√	√	√			√			
Row modulator IDAC				√	√				
Column modulator IDAC				√	√				
Widget Threshold									
Finger threshold	√	√	√	√	√		√	√	√
Noise threshold	√	√	√	√	√	√	√	√	√
Negative noise threshold	√	√	√	√	√	√	√	√	√
Low baseline reset	√	√	√	√	√	√	√	√	√
Hysteresis	√	√	√	√	√	√	√	√	√
ON debounce	√	√	√	√	√	√	√	√	√
Proximity threshold						√			
Touch threshold						√			
Velocity									√
Sensor Parameters									
Compensation IDAC value	√	√	√	√	√	√			
IDAC Values							√	√	√
Selected pins	√	√	√	√	√	√	√	√	√
Position Filter Parameters									
Median filter		√	√		√				√
IIR filter		√	√		√				√
IIR filter coefficient		√	√		√				√
Average filter		√	√		√				√
Jitter filter		√	√		√				√
Adaptive IIR Filter Parameters									
Adaptive IIR filter		√	√		√				√
Position movement threshold		√	√		√				√
Position slow movement threshold		√	√		√				√
Position fast movement threshold		√	√		√				√
IIR coefficient maximum limit		√	√		√				√
IIR coefficient minimum limit		√	√		√				√
IIR coefficient divisor		√	√		√				√
Ballistic Multiplier Parameters									
Acceleration coefficient					√				
Speed coefficient					√				
Divisor value					√				
X-axis speed threshold					√				
Y-axis speed threshold					√				
Centroid Parameters									
Centroid type					√				
Cross-coupling position threshold					√				
Edge correction					√				
Virtual sensor threshold					√				
Penultimate threshold					√				
Two finger detection					√				

Pins Tab

Use the **Pins** tab to assign pins to each sensor. Select the appropriate signal from the pull-down menu.



Note The **Pins** tab is only available if the CapSense Configurator is launched from the Device Configurator.

References

Refer to the following documents for more information, as needed:

- ModusToolbox IDE User Guide
- ModusToolbox Device Configurator Guide
- ModusToolbox CapSense Tuner Guide
- PDL API Reference Guide
- Device Datasheets
- Device Technical Reference Manuals

Version Changes

This section lists and describes the changes for each version of this tool.

Version	Change Descriptions
1.0	New tool.
1.1	Added the Notice List.
	Added more configuration parameters validation.
	Fixed minor issues.

© Cypress Semiconductor Corporation, 2018-2019. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, ModusToolbox, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.