ModusToolbox™ CAPSENSE™ Tuner user guide

ModusToolbox™ tools package version 3.1.0

CAPSENSE™ Tuner version 6.10.0

About this document

Scope and purpose

The CAPSENSE™ Tuner is used to tune CAPSENSE™ applications.

Intended audience

This document helps application developers understand how to use the CAPSENSE™ Tuner as part of creating a ModusToolbox™ application.

Document conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, menus and sub-menus
Italics	Denotes file names and paths.
`Courier New`	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets
File > New	Indicates that a cascading sub-menu opens when you select a menu item

Abbreviations and definitions

The following define the abbreviations and terms used in this document:

- Application – One or more projects related to each other.
- CAPSENSE – capacitive sensing
- Configurator – A GUI-based tool used to configure a resource.
- CSD – self-capacitance sensing method
- CSX – CAPSENSE™ Transmit/Receive (CAPSENSE™ with two electrodes: Tx and Rx), mutual capacitance sensing method
- IDE – integrated development environment
- CSD HW – CAPSENSE™ Sigma Delta - 4th generation hardware (HW) block
- MSC HW – multi sensing converter – 5th generation hardware (HW) block
- MSCLP HW – multi sensing converter low power – 5th generation LP hardware (HW) block
- Peripheral – Any external analog or digital device that provides an input and output for the computer.
- PSoC™ – programmable system-on-chip
- RTT – real time transfer
- SNR – signal-to-noise ratio

- Widget – A CAPSENSE™ functional unit consisting of one sensor or a group of similar sensors, implementing a specific higher-level functionality such as Button, Proximity Sensor, Linear Slider, Radial Slider, Matrix Buttons or Touchpad.

Reference documents

Refer to the following documents for more information as needed:

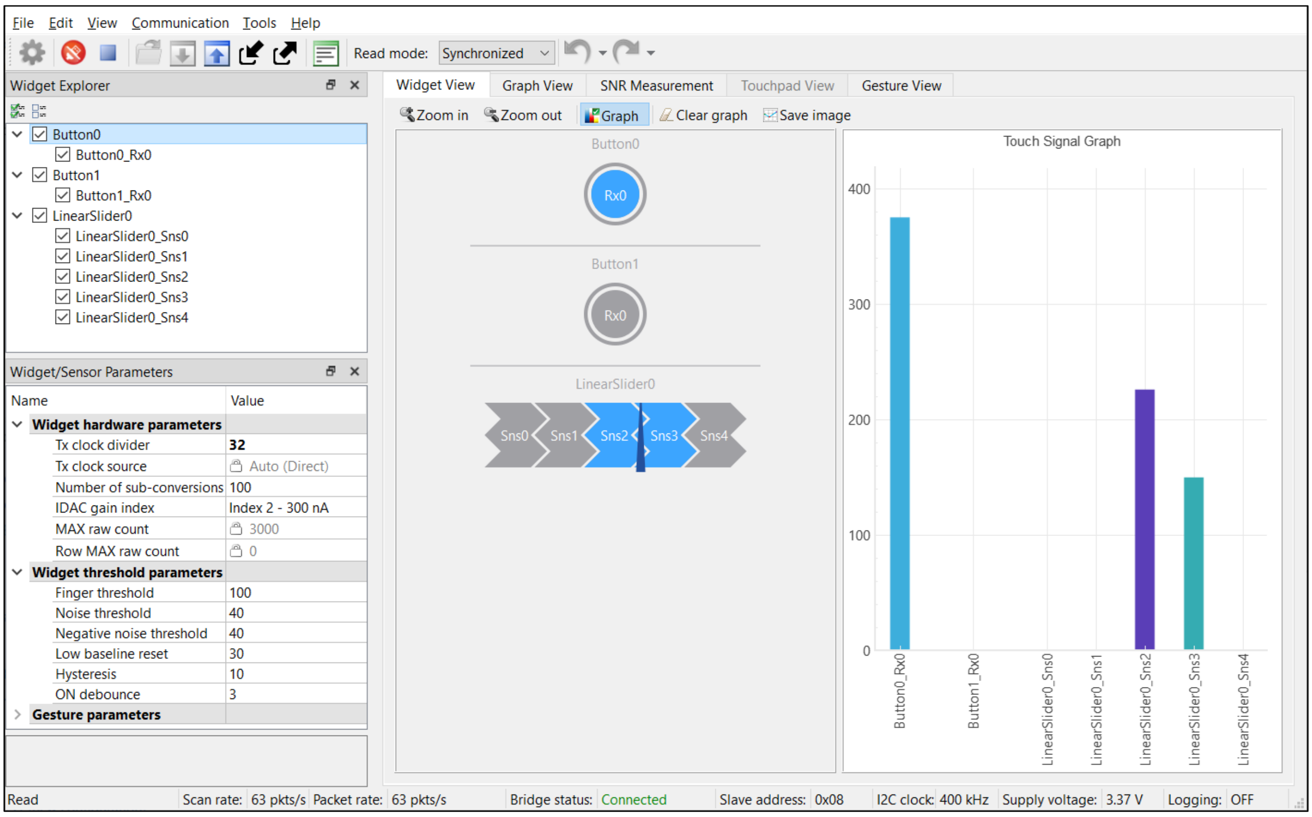
- Eclipse IDE for ModusToolbox™ user guide
- CAPSENSE™ Configurator user guide
- Device Configurator user guide
- MTB CAT1 Peripheral driver library
- MTB CAT2 Peripheral driver library
- Device datasheets
- Device technical reference manuals

Table of contents

1 Overview

The CAPSENSE™ Tuner is a stand-alone tool included with the ModusToolbox™ software. The tool is used to tune CAPSENSE™ applications.

Prior to using the CAPSENSE™ Tuner, create a CAPSENSE™ application and program it into the device. Refer to the CAPSENSE ™ Configurator guide and the CAPSENSE™ middleware documentation for help. Your application must contain the CAPSENSE™ Middleware and a communication interface (I2C, UART or RTT). The Tuner works with the KitProg3, MiniProg4, and UART devices.



Read mode: Synchronized

Widget Explorer:

- ✓ Button0
  - ✓ Button0_Rx0
- ✓ Button1
  - ✓ Button1_Rx0
- ✓ LinearSlider0
  - ✓ LinearSlider0_Sns0
  - ✓ LinearSlider0_Sns1
  - ✓ LinearSlider0_Sns2
  - ✓ LinearSlider0_Sns3
  - ✓ LinearSlider0_Sns4

Widget/Sensor Parameters

Name	Value
Widget hardware parameters	
Tx clock divider	**32**
Tx clock source	Auto (Direct)
Number of sub-conversions	100
IDAC gain index	Index 2 - 300 nA
MAX raw count	3000
Row MAX raw count	0
Widget threshold parameters	
Finger threshold	100
Noise threshold	40
Negative noise threshold	40
Low baseline reset	30
Hysteresis	10
ON debounce	3
Gesture parameters	

Read | Scan rate: 63 pkts/s | Packet rate: 63 pkts/s | Bridge status: Connected | Slave address: 0x08 | I2C clock: 400 kHz | Supply voltage: 3.37 V | Logging: OFF

2 Launch the CAPSENSE™ Tuner

There are several ways to launch the CAPSENSE™ Tuner, and those ways depend on how you use the various tools in ModusToolbox™. However, the easiest way is to launch it using the Device Configurator because the CAPSENSE™ Tuner requires a properly configured CAPSENSE™ application programmed into the device. Refer to the Device Configurator guide for more details.

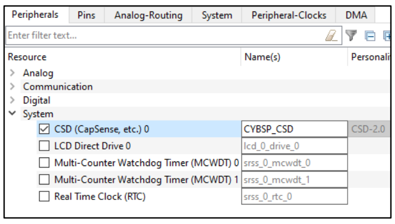
2.1 From the Device Configurator

You can launch the CAPSENSE™ Tuner from the Device Configurator. Refer to the Device Configurator guide for more details.

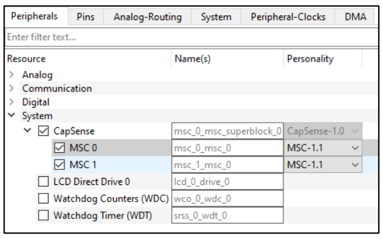
The process differs slightly depending on the hardware block:

- The 4th generation CAPSENSE™ – one CSD resource.
- The 5th generation CAPSENSE™ – two and more MSC resources.
- The 5th generation LP CAPSENSE™ – one MSCLP resource.

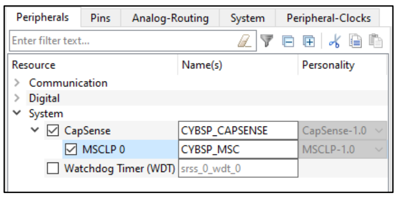
1. On the **Peripherals** tab, select the **CSD (CapSense)** resource or the **CapSense** and one or more **MSC** or **MSCLP** resources, as applicable for your device.



Resource	Name(s)	Personali
Analog		
Communication		
Digital		
System		
✓ CSD (CapSense, etc.) 0	CYBSP_CSD	CSD-2.0
LCD Direct Drive 0	lcd_0_drive_0	
Multi-Counter Watchdog Timer (MCWDT) 0	srss_0_mcwdt_0	
Multi-Counter Watchdog Timer (MCWDT) 1	srss_0_mcwdt_1	
Real Time Clock (RTC)	srss_0_rtc_0	

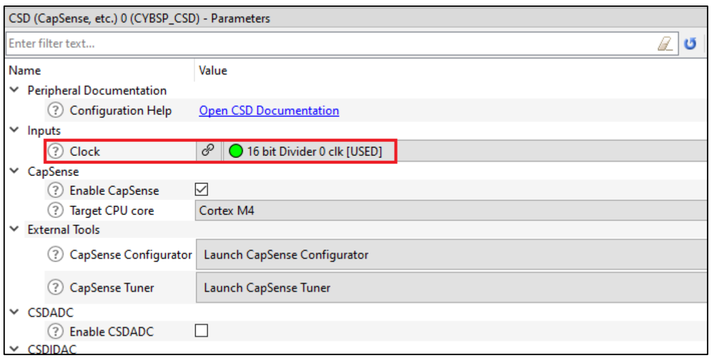


Resource	Name(s)	Personality
Analog		
Communication		
Digital		
System		
✓ CapSense	msc_0_msc_superblock_0	CapSense-1.0
✓ MSC 0	msc_0_msc_0	MSC-1.1
✓ MSC 1	msc_1_msc_0	MSC-1.1
LCD Direct Drive 0	lcd_0_drive_0	
Watchdog Counters (WDC)	wco_0_wdc_0	
Watchdog Timer (WDT)	srss_0_wdt_0	

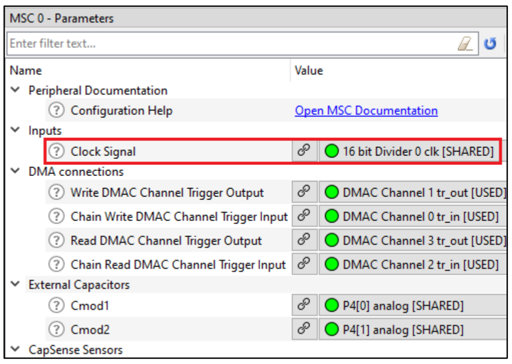


Resource	Name(s)	Personality
Communication		
Digital		
System		
✓ CapSense	CYBSP_CAPSENSE	CapSense-1.0
✓ MSCLP 0	CYBSP_MSC	MSCLP-1.0
Watchdog Timer (WDT)	srss_0_wdt_0	

2. (Skip this step for MSCLP.) On the **Parameters** pane, select an appropriate input **Clock**.

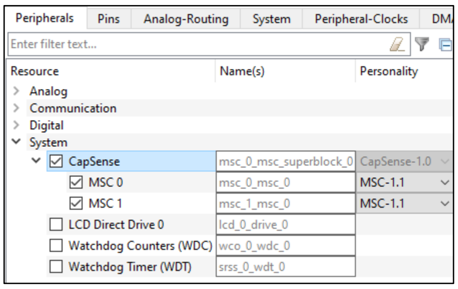


Name	Value
Peripheral Documentation	
Configuration Help	Open CSD Documentation
Inputs	
Clock	16 bit Divider 0 clk [USED]
CapSense	
Enable CapSense	✓
Target CPU core	Cortex M4
External Tools	
CapSense Configurator	Launch CapSense Configurator
CapSense Tuner	Launch CapSense Tuner
CSDADC	
Enable CSDADC	
CSDIDAC	



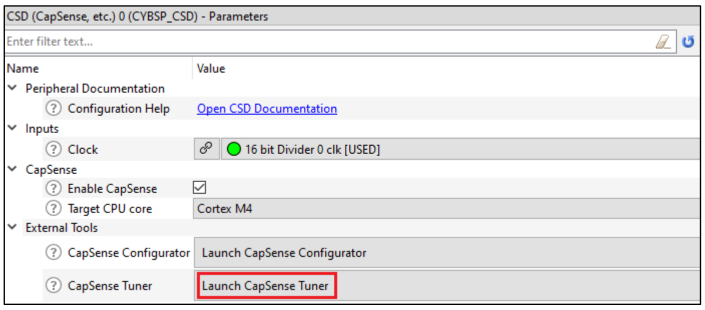
Name	Value
Peripheral Documentation	
Configuration Help	Open MSC Documentation
Inputs	
Clock Signal	16 bit Divider 0 clk [SHARED]
DMA connections	
Write DMAC Channel Trigger Output	DMAC Channel 1 tr_out [USED]
Chain Write DMAC Channel Trigger Input	DMAC Channel 0 tr_in [USED]
Read DMAC Channel Trigger Output	DMAC Channel 3 tr_out [USED]
Chain Read DMAC Channel Trigger Input	DMAC Channel 2 tr_in [USED]
External Capacitors	
Cmod1	P4[0] analog [SHARED]
Cmod2	P4[1] analog [SHARED]
CapSense Sensors	

3. For the MSC and MSCLP HW blocks, select the **CapSense** resource category on the **Peripherals** tab.

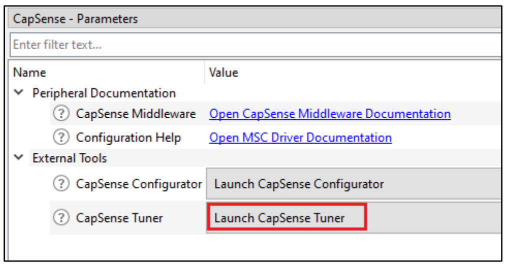


Resource	Name(s)	Personality
Analog		
Communication		
Digital		
System		
☑ CapSense	msc_0_msc_superblock_0	CapSense-1.0
☑ MSC 0	msc_0_msc_0	MSC-1.1
☑ MSC 1	msc_1_msc_0	MSC-1.1
☐ LCD Direct Drive 0	lcd_0_drive_0	
☐ Watchdog Counters (WDC)	wco_0_wdc_0	
☐ Watchdog Timer (WDT)	srss_0_wdt_0	

4. On the **Parameters** pane, click the **Launch CapSense Tuner** button.



Name	Value
Peripheral Documentation	
Configuration Help	Open CSD Documentation
Inputs	
Clock	🟢 16 bit Divider 0 clk [USED]
CapSense	
Enable CapSense	☑
Target CPU core	Cortex M4
External Tools	
CapSense Configurator	Launch CapSense Configurator
CapSense Tuner	Launch CapSense Tuner



Name	Value
Peripheral Documentation	
CapSense Middleware	Open CapSense Middleware Documentation
Configuration Help	Open MSC Driver Documentation
External Tools	
CapSense Configurator	Launch CapSense Configurator
CapSense Tuner	Launch CapSense Tuner

2.2 make command

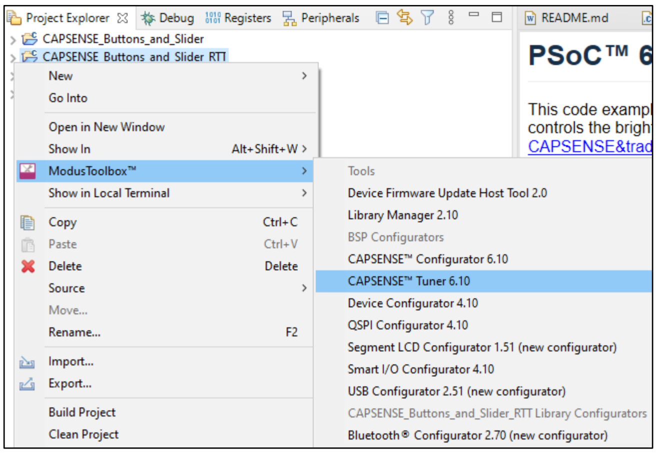
As described in the ModusToolbox™ user guide Build System chapter, you can run numerous `make` commands in the application directory, such as launching the CAPSENSE™ Tuner. After you have created a ModusToolbox™ application, navigate to the application directory and type the following command in the appropriate bash terminal window:

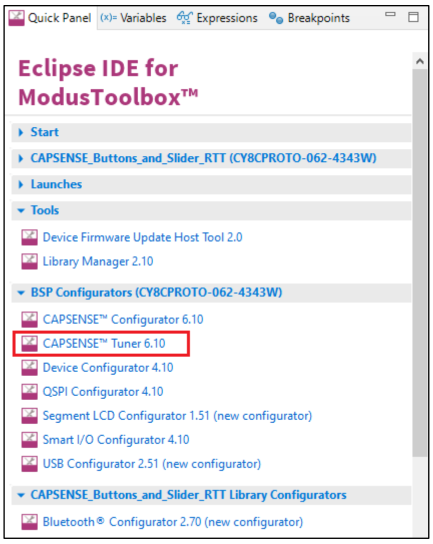
```
make capsense-tuner
```

This command opens the CAPSENSE™ Tuner GUI for the specific application in which you are working.

2.3 Eclipse IDE

If the selected application already includes a configuration (*design.cycapsense*) file, you can launch the CAPSENSE™ Tuner from the Eclipse IDE. Right-click on the project in the Project Explorer and select **ModusToolbox™ > CAPSENSE™ Tuner <version>**. You can also click the **CAPSENSE™ Tuner** link in the IDE Quick Panel. Refer to the Eclipse IDE for ModusToolbox™ user guide and CAPSENSE™ Configurator user guide for more details.




Quick Panel
Variables
Expressions
Breakpoints
Eclipse IDE for ModusToolbox™
Start
CAPSENSE_Buttons_and_Slider_RTT (CY8CPROTO-062-4343W)
Launches
Tools
Device Firmware Update Host Tool 2.0
Library Manager 2.10
BSP Configurators (CY8CPROTO-062-4343W)
CAPSENSE™ Configurator 6.10
CAPSENSE™ Tuner 6.10
Device Configurator 4.10
QSPI Configurator 4.10
Segment LCD Configurator 1.51 (new configurator)
Smart I/O Configurator 4.10
USB Configurator 2.51 (new configurator)
CAPSENSE_Buttons_and_Slider_RTT Library Configurators
Bluetooth® Configurator 2.70 (new configurator)

Note: *These steps are the minimum required to launch the CAPSENSE™ Tuner. However, the Tuner will not work without a properly configured CAPSENSE™ application programmed into the device.*

2.4 Executable (GUI)

You can launch the CAPSENSE™ Tuner GUI by running its executable as appropriate for your operating system. By default, the CAPSENSE™ Tuner is installed here:

<install_dir>/ModusToolbox/tools_<version>/capsense-configurator

When run independently, the application opens without any information. You need to open a specific CAPSENSE™ application configuration file that was created by the CAPSENSE™ Configurator for the Tuner to work. See Menus for more information about opening a configuration file.

2.5 From the Command Line

You can run the Tuner from the command line. However, there are few reasons to do this in practice.

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

3 Quick start

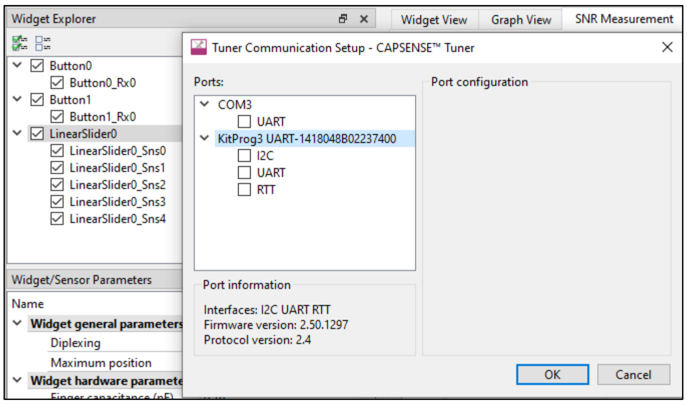
This section provides a simple workflow for how to use the CAPSENSE™ Tuner.

1. Create a CAPSENSE™ application with a tuner communication interface, and program the application into the device. Refer to the Eclipse IDE for ModusToolbox™ user guide and the CAPSENSE™ Configurator user guide for more details.

Note: *The simplest way to start with CAPSENSE™ Tuner is to use the PSoC™ 6 CAPSENSE™ Buttons and Slider code example configured to work on PSoC™ 6 MCU kits or PSoC™ 4 MSC CAPSENSE™ CSD Button Tuning to work on PSoC™ 4 MCU kits.*

2. Launch the CAPSENSE™ Tuner.

3. Start Communication. Click **Connect** and select the required communication interface.



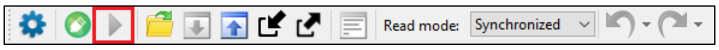
Port information

Interfaces: I2C UART RTT
Firmware version: 2.50.1297
Protocol version: 2.4

Note: *Refer to the KitProg user guide for the supported configurations and modes. For this example, I2C address, sub-address, and speed must match the configuration.*

Note: *For the I2C communication, the Tuner only supports the 2-Bytes Sub-address.*

5. Click **Start** to extract data.

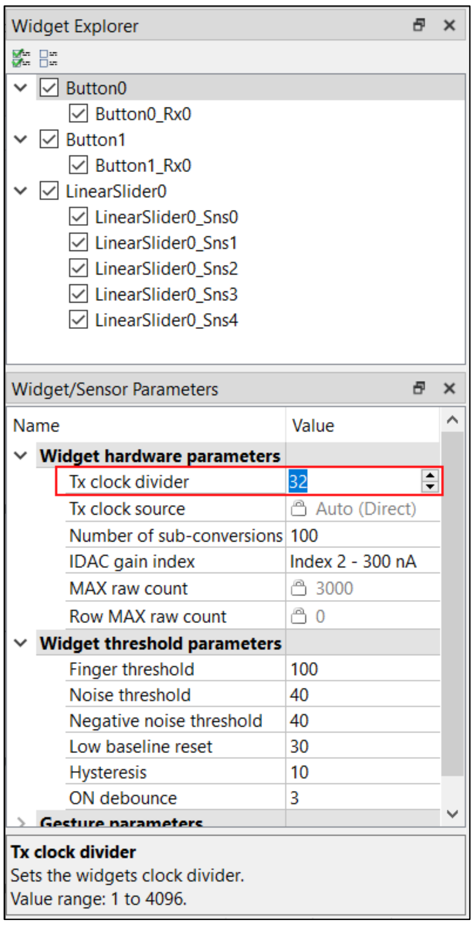


Read mode: Synchronized

6. Touch the sensors on the hardware and notice the change in the sensor/widget status on Widget View Tab.

7. Open the **Graph View** tab. Check the sensors in the Widget Explorer Pane to observe sensor signals in the graph. Touch the sensors and notice the signal change on the Graph View Tab.

8. Change widget/sensor parameter values as needed. Then, apply the new settings to the device using the **Apply to Device** button.


Widget Explorer

- Button0
  - Button0_Rx0
- Button1
  - Button1_Rx0
- LinearSlider0
  - LinearSlider0_Sns0
  - LinearSlider0_Sns1
  - LinearSlider0_Sns2
  - LinearSlider0_Sns3
  - LinearSlider0_Sns4

Widget/Sensor Parameters

Name	Value
Widget hardware parameters	
Tx clock divider	32
Tx clock source	Auto (Direct)
Number of sub-conversions	100
IDAC gain index	Index 2 - 300 nA
MAX raw count	3000
Row MAX raw count	0
Widget threshold parameters	
Finger threshold	100
Noise threshold	40
Negative noise threshold	40
Low baseline reset	30
Hysteresis	10
ON debounce	3
Gesture parameters	

Tx clock divider
Sets the widgets clock divider.
Value range: 1 to 4096.

You can do this when using Manual or SmartSense (Hardware parameters only) modes for tuning:

- To edit the threshold parameters, use SmartSense (Hardware parameters only) mode.
- To edit all the parameters, use Manual mode.
- When SmartSense (Full Auto-Tune) is selected for CSD tuning mode, parameters are read-only (except the Finger Capacitance parameter).

9. Save the Tuner parameters. Click the **Apply to Project** button.
10. Exit the Tuner application.

4 GUI description

The CAPSENSE™ Tuner application contains menus, a toolbar, panes, tabs, and a status bar, all used to tune a CAPSENSE™ application.

4.1 Menus

4.1.1 File

- **Open…** – Opens a <file_name>.cycapsense configuration file. This command is visible only when running the Tuner independently from the Eclipse IDE.
- **Apply to Device** – Commits changed values of a widget/sensor parameter to the device. This command becomes active if a value of any configuration parameter from the Tuner application changes (that is, if the parameter values in the Tuner and the device are different). This is an indication to apply the changed parameter values to the device. For the Simplex UART interface this command is grayed out.
- **Apply to Project** – Commits changed values of a widget / sensor parameter to the CAPSENSE™ project.
- **Open in System Explorer** – This opens your computer's file explorer tool to the folder that contains the *.cycapsense file.
- **Import…** – Imports a specified configuration file.
- **Export…** – Exports the current configuration file into a specified file.
- **Export Register Maps to PDF…** – Exports the current configuration register map in PDF format.
- Recent files – Shows recent files that you can open directly.
- **Exit** – Asks to save changes if there are any and closes the Tuner. Changes are saved to the configuration file.

4.1.2 Edit

- **Undo** – Undoes the last action or sequence of actions.
- **Redo** – Redoes the last undone action or sequence of undone actions.

4.1.3 View

- **Widget Explorer** – Hides or shows the Widget Explorer pane where widgets and sensors tree display.
- **Widget/Sensor Parameters** – Hides or shows the Widget/Sensor Parameters pane.
- **OpenOCD Log** – Hides or shows the OpenOCD Log pane.
- **Gesture Event History** – Logs the detected gestures information. Displays in the View when any gesture is available.
- **Gesture Monitor** – Provides visual indication for a detected gesture. Displays in the View when any gesture is available
- **Toolbar** – Hides or shows the Toolbar. Enabled by default.
- **Reset View** – Resets the view to the default.

4.1.4 Communication

- **Connect** – Connects to the device via a communication channel selected in the Tuner Communication Setup dialog. If the channel was not previously selected, the Tuner Communication dialog is shown.
- **Disconnect** – Closes the communication channel with the connected device.

- **Start** – Starts reading data from the device.
- **Stop** – Stops reading data from the device.

4.1.5 Tools

- **Tuner Communication Setup…** – Opens the configuration dialog to set up a communication channel with the device.
- **Options…** – Opens the configuration dialog to set up Tuner preferences: Display, SNR, Logging, Advanced.

4.1.6 Help

- **View Help** – Opens the CAPSENSE™ Tuner guide (this document).
- **About CAPSENSE™ Tuner** – Opens the About box to display the version information, with links to open https://www.infineon.com and the current session log file.

4.2 Toolbar

Contains frequently used buttons that duplicate the main menu items:

- ⚙ Opens the configuration dialog to set up a communication channel with the device. The same as the **Tools > Tuner Communication Setup** menu command.
- Connects to the device via a communication channel selected in the Tuner Communication Setup dialog. The same as the **Communication > Connect** menu command.
- Closes the communication channel with the connected device. The same as the **Communication > Disconnect** menu command
- ▶ Starts reading data from the device. The same as the **Communication > Start** menu command.
- ■ Stops reading data from the device. The same as the **Communication > Stop** menu command.
- Opens a configuration file. The same as the **File > Open** menu command.
- Commits changed values of a widget/sensor parameter to the device. The same as the **File > Apply to Device** menu command.
- Commits changed values of a widget / sensor parameter to the CAPSENSE™ project. The same as the **File > Apply to Project** menu command.
- Imports a specified configuration file. The same as the **Import** menu command.
- Exports the current configuration file into a specified file. The same as the **Export** menu command.
- Starts or stops data logging into a specified file.
- **Read mode** – Selects the Tuner communication mode with a device (I2C).
- ↶ Undoes the last action or sequence of actions. The same as the **Edit > Undo** menu command.

- Redoes the undone action or sequence of undone actions. The same as the **Edit > Redo** menu command.

4.3 Widget Explorer pane

The Widget Explorer pane contains a tree of widgets and sensors used in the CAPSENSE™ application. You can expand/collapse the Widget nodes to show/hide widget's sensor nodes. You can check/uncheck individual widgets and sensors in the Widget Explorer pane. The widget checked status affects its visibility in the *Widget View* and the position graph series in the *Graph View* while the sensor checked status controls the visibility of the sensor raw count / baseline / signal / status graph series in the *Widget View* and signals in the *Touch Signal Graph* on the *Widget View*.

Selecting a widget or sensor in the Widget Explorer pane updates the selection in the Widget/Sensor Parameters pane.

Note: For CSX widgets, the sensor tree displays individual nodes (Rx0_Tx0, Rx0_Tx1 …) contrary to the CAPSENSE™ Configurator where the CSX electrodes are displayed (Rx0, Rx1 … Tx0, Tx1 ...).

Note: For MSC HW configurations with enabled Multi-frequency mode, there is a Median checkbox in the Widget Explorer for each sensor that shows a set of F0 values received from a kit in the Graph View tab.

4.4 Widget/Sensor parameters pane

The Widget/Sensor parameters pane displays the parameters of a widget or sensor selected in the Widget Explorer tree. The grid is similar to the grid on the Widget Details tab in the CAPSENSE™ Configurator. The main difference is that some parameters are available for modification in the Configurator, but not in the Tuner. When a parameter is selected, its description displays on the panel below the parameters list.

This pane includes the following parameters:

- Widget General Parameters – Cannot be modified from the Tuner because corresponding parameter values reside in the application flash widget structures that cannot be modified at runtime.
- Widget Hardware Parameters – Cannot be modified for the CSD widgets when CSD tuning mode is set to SmartSense (Full Auto-Tune) or SmartSense (Hardware parameters only) in the CAPSENSE™ Configurator. In Manual tuning mode (for all widgets), any change to Widget Hardware Parameters requires hardware re-initialization performed only if the Tuner communicates with the device in Synchronized mode.
- Widget Threshold Parameters – Cannot be modified for the CSD widgets when CSD tuning mode is set to SmartSense (Full Auto-Tune) in the Configurator. In Manual tuning mode (for all widgets), threshold parameters are always writable (Synchronized mode is not required). The exception is the ON debounce parameter that also requires hardware re-initialization (in the same way as the hardware parameters).
- Sensor Parameters – Sensor-specific parameters. The Tuner application displays only Compensation IDAC or CDAC values. When the IDAC / CDAC auto-calibration is enabled, the parameter is read-only and displays values as calibrated by the firmware. When the auto-calibration is disabled, the IDAC / CDAC value entered in the CAPSENSE™ Configurator displays, and the parameter becomes writable in Synchronized mode.
- Filter Parameters and Centroid Parameters – Cannot be modified at runtime from the Tuner because, unlike the other parameters, these parameter values reside in the application flash widget structures that cannot be modified at runtime.
- Gesture Parameters – Synchronized communication mode must be selected to update the Gesture parameters during runtime from the Tuner application.

Note: The position filter parameters, Adaptive IIR filter parameters, and Gesture parameters reflect data stored in a loaded configuration file. Actual values may vary on a target device connected to the CAPSENSE™ Tuner.

4.5 Tabs

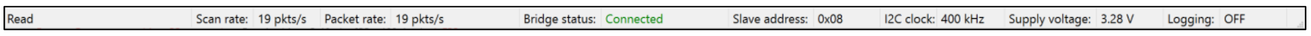
The application consists of the following tabs:

- Widget View – Displays the widgets, their touch status, and the touch signal bar graph.
- Graph View – Displays the sensor data charts.
- SNR Measurement – Provides the SNR measurement functionality.
- Touchpad View – Displays the touchpad heatmap.
- Gesture View – Displays the Gesture operation.

4.6 Status bar

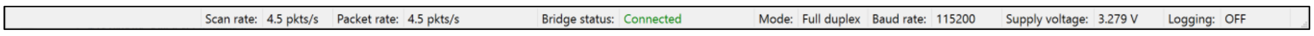
The status bar at the bottom of the GUI displays information related to the communication state between the Tuner and the device. Some sections differ depending on the communication type as follows:

I2C interface



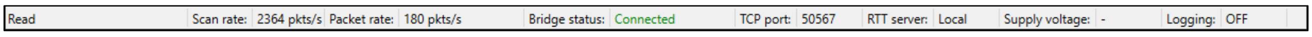
```
Read    Scan rate: 19 pkts/s    Packet rate: 19 pkts/s    Bridge status: Connected    Slave address: 0x08    I2C clock: 400 kHz    Supply voltage: 3.28 V    Logging: OFF
```

UART interface



```
Scan rate: 4.5 pkts/s    Packet rate: 4.5 pkts/s    Bridge status: Connected    Mode: Full duplex    Baud rate: 115200    Supply voltage: 3.279 V    Logging: OFF
```

RTT interface



```
Read    Scan rate: 2364 pkts/s    Packet rate: 180 pkts/s    Bridge status: Connected    TCP port: 50567    RTT server: Local    Supply voltage: -    Logging: OFF
```

- **Scan rate** – A number of scans performed by the device per second.
- **Packet rate** – A number of read samples performed by the Tuner per second. The count depends on multiple factors: the selected communication channel, communication speed, and amount of time for a single scan. The **Packet rate** can be lower or equal to the **Scan rate**.
- **Bridge status** – Either Connected, when the communication channel is active, or Disconnected otherwise.
- **Slave address** (I2C) – The address of the I2C slave configured for the current communication channel.
- **I2C clock** (I2C) – The data rate used by the I2C communication channel.
- **Mode** (UART) – The UART data transmission mode.
- **Baud rate** (UART) – The data rate, at which CAPSENSE™ operates with the current settings.
- **Tuner TCP port** (RTT) – Used by the Tuner to connect to the RTT server.
- **RTT Server** (RTT) – Indicates if the RTT server is started locally or on a specific host.
- **Supply voltage** – The supply voltage.
- **Logging** – Either ON (when the data logging to a file in progress) or OFF.

5 Tuner communication setup

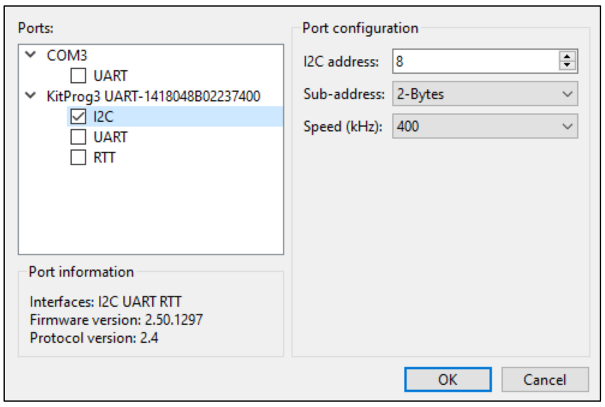
The **Tuner Communication Setup** dialog is used to establish communication between the Tuner and target device. The Tuner supports I2C, UART, and RTT communication interfaces.

Select **Tools > Tuner Communication Setup…** on the menu to open the dialog or click the **Tuner Communication Setup** button. The dialog opens automatically after clicking the **Connect** button if no device was previously selected.

Select the device and communication interface by checking the item with the interface name. Change the interface configuration parameters to match the configuration of the communication peripheral in the application.

Note: *The parameters in the Tuner Communication Setup dialog must be identical to the parameters of the EZI2C or UART driver in the application.*

5.1 I2C



Port information

Interfaces: I2C UART RTT
Firmware version: 2.50.1297
Protocol version: 2.4

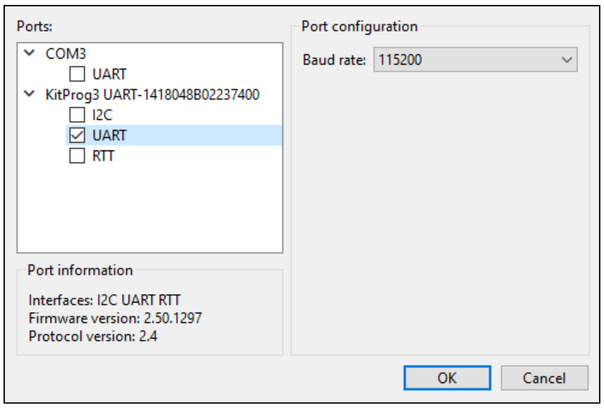
I2C address: 8
Sub-address: 2-Bytes
Speed (kHz): 400

To establish the I2C communication between the Tuner and target device, configure the Tuner communication parameters to match the parameters of the device EzI2C settings.

The I2C communication can work in two read modes:

- **Synchronized** –Application firmware periodically calls a corresponding Tuner function: Cy_CapSense_RunTuner(). The Tuner synchronizes data reading and firmware execution to preserve data coherency. CAPSENSE™ middleware waits for the Tuner read/write operation to be completed prior to execution of sensor scan and processing tasks. The Tuner does not read/write data until CAPSENSE™ middleware completes scanning and data processing tasks for all widgets in the application. The SNR and noise measurements are most accurate, and most tuning parameters can be edited in real time updating in this mode.
- **Asynchronized** – The Tuner reads data asynchronously with sensor scanning and data processing. Due to this, data coherency may be corrupted. So, the Tuner may read only partially updated sensor data. For example, the device completed scanning of only the first sensor in a row. At this moment, the Tuner reads data of the latest scan from the first sensor and data of previous scans from the remaining sensor. This can occur to all sensors, when the Tuner reads data prior to the completion of data processing tasks, such as baseline update and filter execution. SNR and noise measurements are less accurate due to non-coherent data reading and only a limited set of tuning parameters can be edited in real time in this mode.
  Asynchronized mode may work unreliably with low power widgets configurations when the application is mostly in Deep Sleep. The Tuner may miss the time window when the device awakes and is able to communicate with the Tuner.

5.2 UART



Baud rate: 115200

Port information

Interfaces: I2C UART RTT
Firmware version: 2.50.1297
Protocol version: 2.4

To establish the UART communication between the Tuner and target device, configure the Tuner communication parameters to match the parameters of the device UART settings.

Depending on the application capabilities, the UART communication may work in two modes:

- Full duplex mode – Provides data transmission in two reverse directions: target device → Tuner→ target device. You can change the parameters of the widgets/sensors at runtime. To upload the modified configuration, click the **Apply to Device** button.
- Simplex mode – Provides data transmission in one direction: target device → Tuner. You cannot upload the parameters of the widgets/sensors to the device at runtime.

Note: *Recommended – enable the UART Flow control on the target device for bitrates equal or higher than 1000000.*

5.3 RTT

The Real Time Transfer (RTT) is an interface specified by SEGGER based on basic memory reads and writes to transfer data bidirectionally between the target and host. Before configuring the RTT setup dialog, it is recommended to familiarize yourself with the RTT protocol fundamentals at https://wiki.segger.com/RTT.

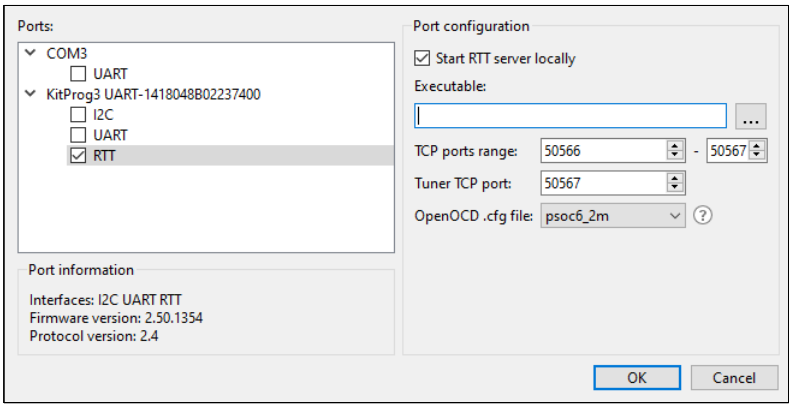
The data transfer between the host and target devices is organized through channels. The Tuner communicates to the server (OpenOCD) via the TCP/IP protocol. The server in turn uses the SWD interface to communicate with the device.

RTT communication advantages:

- SWD pins are used instead of reserving EZI2C or UART pins.
- The communication speed is the highest among all options.

To establish the RTT communication between the Tuner and target device, start either the local RTT server or connect to an existing one. For details on the application layer, refer to Section 5.3.3.

5.3.1 Start RTT server locally


Ports:
- COM3
  - ☐ UART
- KitProg3 UART-1418048B02237400
  - ☐ I2C
  - ☐ UART
  - ☑ RTT

Port information

Interfaces: I2C UART RTT
Firmware version: 2.50.1354
Protocol version: 2.4

Port configuration

☑ Start RTT server locally

Executable:

TCP ports range: 50566 - 50567

Tuner TCP port: 50567

OpenOCD .cfg file: psoc6_2m

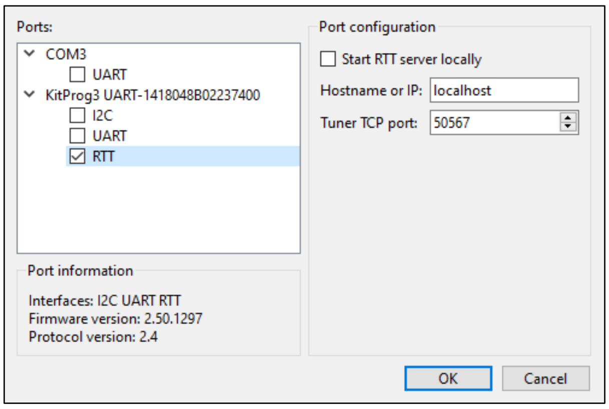
To start the RTT server in your local environment, select **Start RTT server locally** and modify the following fields:

- **Executable** – The .elf file generated by the firmware build. In most cases, it will be prefilled automatically based on the ModusToolbox™ project.
- **TCP ports range** – Used for the RTT communication. Each port is mapped to a channel. Set up at least two channels because Channel 0 has limitations.
- **Tuner TCP port** – The TCP port used by the Tuner.
- **OpenOCD .cfg file** – Provides the .cfg file name for OpenOCD. In most cases, it will be detected automatically based on the application target device and will not display on the GUI. If becomes visible when the .cfg file name value cannot be detected by the Tuner automatically and must be selected manually. The .cfg file name can be found in the .launch file of your project.

To manage the RTT server, the Tuner uses the **rtt.tcl** file, which can be found along with the Tuner executable (or capsense-tuner.app/Contents/Resources/ on macOS). It starts an OpenOCD session, which lasts while the Tuner is connected to the device.

Note: Only one instance of OpenOCD can run at a time. If you want to debug the application simultaneously, refer to RTT communication with simultaneous debug.

5.3.2 Connect to an existing RTT server


Ports:
- COM3
  - ☐ UART
- KitProg3 UART-1418048B02237400
  - ☐ I2C
  - ☐ UART
  - ☑ RTT

Port information

Interfaces: I2C UART RTT
Firmware version: 2.50.1297
Protocol version: 2.4

Port configuration

☐ Start RTT server locally

Hostname or IP: localhost

Tuner TCP port: 50567

To connect to an existing RTT server, unselect **Start RTT server locally** and modify the following fields.

- **Hostname or IP** – Specifies the host or IP address of the running RTT server.
- **Tuner TCP port** – The TCP port used by the Tuner.

5.3.3 Setting up RTT communication in the application

To enable the RTT communication, include the SEGGER RTT library in the application. It is a part of the J-Link Software and Documentation Pack, which is available for download on the SEGGER official website. The RTT library is located inside the "Samples" folder. Place it in the project root folder. Refer to the RTT API documentation to learn how to use it.

Sample code must be added to the *main.c* file:

- Include a header

  ```
  #include "SEGGER_RTT/RTT/SEGGER_RTT.h"
  ```

- Initialize RTT

  ```
  SEGGER_RTT_Init();
  ```

  Configure the up and down buffers.
  The size of the up buffer should be at least sizeof(cy_capsense_tuner) + 5 + 1 (5 bytes for the packet header and trailer, 1 byte for internal RTT purposes).
  The size of the down buffer should be at least 32 bytes.

  ```
  SEGGER_RTT_ConfigUpBuffer(RTT_TUNER_CHANNEL, "tuner", &tunerUpBuf, sizeof(tunerUpBuf),
  SEGGER_RTT_MODE_NO_BLOCK_SKIP);
  SEGGER_RTT_ConfigDownBuffer(RTT_TUNER_CHANNEL, "tuner", &tunerDownBuf, sizeof(tunerDownBuf),
  SEGGER_RTT_MODE_BLOCK_IF_FIFO_FULL);
  ```

- Implement the send and receive functions. Assign them to the callbacks.

  ```
  static void rtt_tuner_send(void * context);
  static void rtt_tuner_receive(uint8_t ** packet, uint8_t ** tunerPacket, void * context);

  ...

  cy_capsense_context.ptrInternalContext->ptrTunerSendCallback    = rtt_tuner_send;
  cy_capsense_context.ptrInternalContext->ptrTunerReceiveCallback = rtt_tuner_receive;

  ...

  void rtt_tuner_send(void * context)
  {
      (void)context;
      /* Packet size including 2-byte header and 3-byte trailer */
      uint16_t elemCount = sizeof(cy_capsense_tuner) + 5;

      /* Lock RTT */
      SEGGER_RTT_LOCK();
      SEGGER_RTT_BUFFER_UP *buffer = _SEGGER_RTT.aUp + RTT_TUNER_CHANNEL;

      /* Copy CAPSENSE tuner structure data to the buffer */
      ...

      /* Manually update the pointers */
      buffer->RdOff = 0;
      buffer->WrOff = elemCount;

      /* Unlock RTT */
      SEGGER_RTT_UNLOCK();
  }

  ...

  void rtt_tuner_receive(uint8_t ** packet, uint8_t ** tunerPacket, void * context)
  {
      while(0 != SEGGER_RTT_HasData(RTT_TUNER_CHANNEL))
  ```

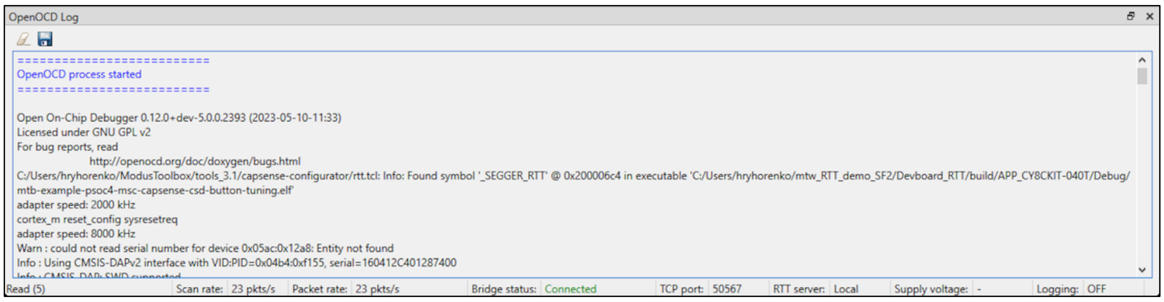
```
    {
        uint8_t byte;
        SEGGER_RTT_Read(RTT_TUNER_CHANNEL, &byte, 1);
        ...
    }
}
```

The Tuner is expected to receive packets in the following format:

```
OD 0A <cy_capsense_tuner> 00 FF FF
```

5.3.4 OpenOCD Log pane

The Tuner offers a special OpenOCD Log view to observe OpenOCD communication logs.



```
==========================
OpenOCD process started
==========================

Open On-Chip Debugger 0.12.0+dev-5.0.0.2393 (2023-05-10-11:33)
Licensed under GNU GPL v2
For bug reports, read
          http://openocd.org/doc/doxygen/bugs.html
C:/Users/hryhorenko/ModusToolbox/tools_3.1/capsense-configurator/rtt.tcl: Info: Found symbol '_SEGGER_RTT' @ 0x200006c4 in executable 'C:/Users/hryhorenko/mtw_RTT_demo_SF2/Devboard_RTT/build/APP_CY8CKIT-040T/Debug/
mtb-example-psoc4-msc-capsense-csd-button-tuning.elf'
adapter speed: 2000 kHz
cortex_m reset_config sysresetreq
adapter speed: 8000 kHz
Warn : could not read serial number for device 0x05ac:0x12a8: Entity not found
Info : Using CMSIS-DAPv2 interface with VID:PID=0x04b4:0xf155, serial=160412C401287400
Info : CMSIS-DAP: SWD supported
```

Read (5) | Scan rate: 23 pkts/s | Packet rate: 23 pkts/s | Bridge status: Connected | TCP port: 50567 | RTT server: Local | Supply voltage: - | Logging: OFF

5.3.5 RTT communication with simultaneous debug

The RTT interface allows communication with the CAPSENSE™ Tuner and debugging simultaneously over the same pair of SWD pins. The RTT communication utilizes OpenOCD. To debug simultaneously, establish a connection with the Tuner and then attach the debugger to the already running OpenOCD session. For details, refer to the debug section of your IDE user guide.

5.4 Communication and Low power mode

The Tuner requires a working communication channel (I2C, UART, RTT). While the device is operating in Low power mode, it cannot communicate with the Tuner. In order to have a stable communication, Synchronization mode was introduced between the Tuner and the device. In this mode, the Tuner controls the device, particularly the moment of starting a next scan. If this process is interrupted, communication may be lost. To avoid this, follow the recommendations:

These are the limitations necessary for maintaining the successful connection of the Tuner with a device:

- The PSoC™ side firmware ensures that after the device exits Deep Sleep mode, the device stays in the active state with the working communication channel, which enables the Tuner to receive at least one packet with valid data.
- The device sleeps less than the Tuner-defined timeout (see the LP command timeout in the Tuner Configuration Options dialog, Advanced tab).
- Implement the Deep Sleep callback, which waits for the completion of an EzI2C/UART transaction before the transition to Deep Sleep.

Also recommended:

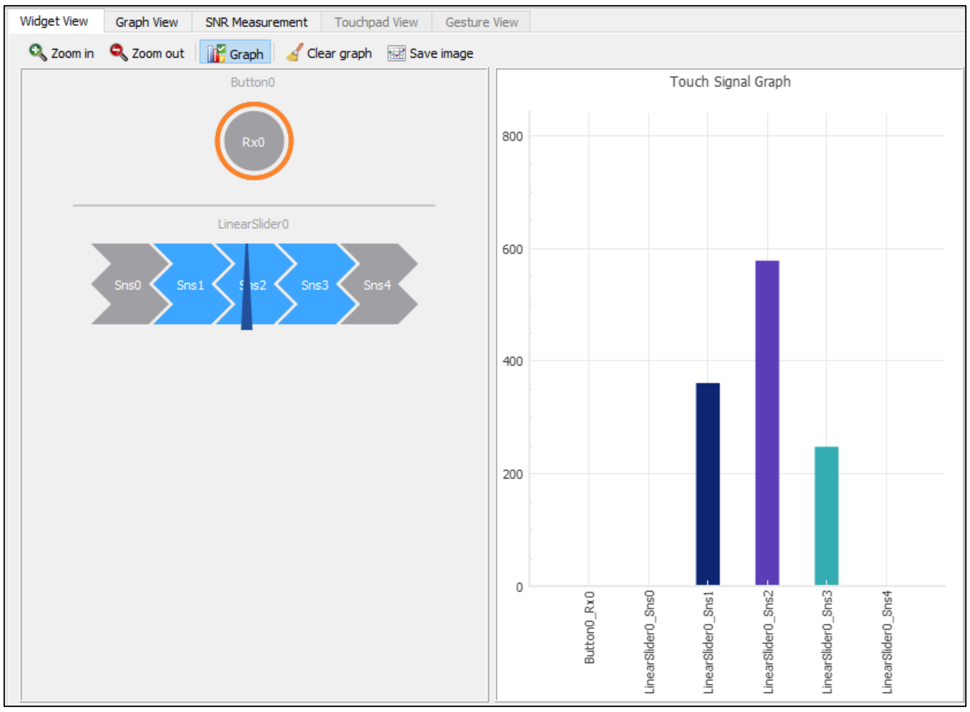
- Add Cy_SysLib_Delay 100-500 ms after the CAPSENSE™ initialization completes before the transition to Deep Sleep to give the Tuner time to exchange packets with the device.

The Tuner has the following timeouts by default:

- Connect timeout: 3 sec
- Read/write timeout: 250 sec.

6 Widget View tab

The **Widget View** provides a visual representation of all widgets selected in the Widget Explorer Pane. If a widget consists of more than one sensor, individual selected sensors are highlighted in the Widget Explorer Pane and Widget/Sensor Parameters Pane.



The Widget and/or sensors are highlighted when the device reports their touch status as active.

Some additional features are available depending on the widget type.

6.1 Toolbar

The toolbar provides quick access to different configuration options that affect the Widget View display.

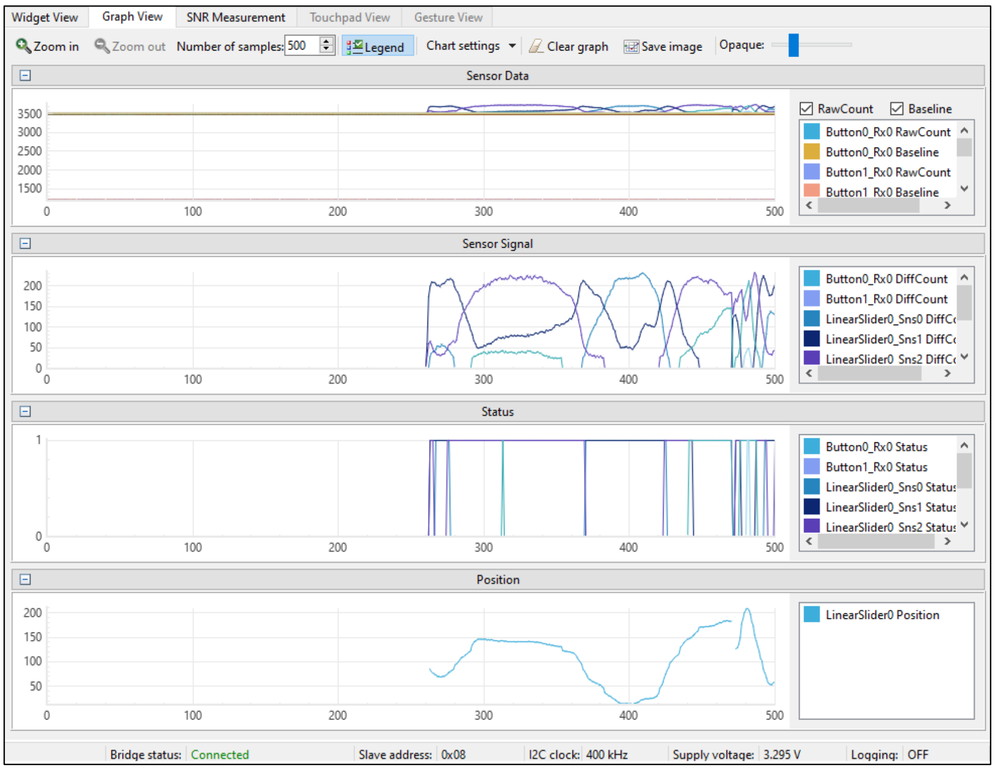
- Zoom in – Zooms in the widgets. Press the [**Esc**] key to undo zoom.
- Zoom out – Zooms out the widgets. Press the [**Esc**] key to undo zoom.
- Graph – Shows or hides the Touch Signal Graph.
- Clear graph – Clears the Touch Signal Graph.
- Save image – Opens the dialog to save the Widget View tab as an image. To save the whole main window as an image, press and hold the [**Ctrl**] key. The supported formats: .PNG, .JPG, BMP.

6.2 Touch Signal Graph

The Widget view also displays the Touch Signal Graph. This graph contains a touch signal level for each sensor selected in the Widget Explorer Pane.

7 Graph View tab

The Graph View displays graphs for the sensors selected in the Widget Explorer Pane.



The following graphs are available:

- Sensor Data – Displays RawCount and Baseline. Click a corresponding check box to see them. Under Multi-frequency mode, RawCount and Baseline are available for three channels: 0, 1, 2.
- Sensor Signal – Displays signal differences.
- Status – Displays a sensor status (Touch/No Touch).
- Position – Displays touch positions for the Linear Slider, Radial Slider, and Touchpad widgets.

Note: *For 5th generation CAPSENSE™ configurations with enabled Multi-frequency mode, there is a Median graph on the Sensor Signal graph for each sensor.*
For Multi-frequency mode, three varieties are scanned for each sensor at three different frequencies and three different raw-counts are obtained for the same sensor. The sensor state is estimated by one median value calculated from the obtained three raw-counts.
A Median graph shows a set of F0 values received from a kit. The graph becomes visible if at least one of the graphs F0, F1, F2 is visible. Then, the Tuner calculates the actual value of F0 from received data.

7.1 Toolbar

The toolbar provides quick access to different configuration options that affect the Graph View display.

- Zoom in – Zooms in all graphs synchronously along the X axis. Press the [**Esc**] key to undo zoom.
- Zoom out – Zooms out all graphs synchronously along the X axis. Press the [**Esc**] key to undo zoom.
- Number of samples – Defines the total amount of data samples shown on a single graph.
- Legend – Shows or hides the sensor series descriptions (with names and colors) in graphs.

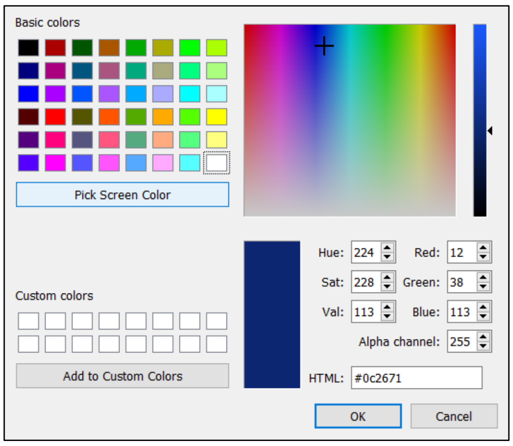
- Chart settings – Opens the menu for editing the chart line series parameters.
- Clear graph – Clears all graphs.
- Save image – Opens a dialog to save the Graph View tab as an image. To save the whole main window as an image, press and hold the [**Ctrl**] key. The supported formats: .PNG, .JPG, .BMP.
- Opaque – The level of opacity of the area below the line series.

7.2 Graph functionality

The Graph functionality includes:

- **Highlighting** – If you click a sensor line series on the graph, the corresponding sensor series highlights on the Legend pane. Likewise, if you click a sensor series on the Legend pane, it highlights a sensor line series on the graph. To highlight lines for all graphs at the same time, press and hold the [**Ctrl**] key. To select multiple sensors in the other View tabs, press and hold the [**Alt**] key.
- **Pan** and **Zoom** – Allow you to examine graphs in more detail. Use a mouse drag for pan. Press and hold the [**Shift**] key to select the area you need to examine. Press and hold the [**Ctrl**] key to examine all the graphs simultaneously.

  Use the mouse wheel for Zoom. The [**Ctrl**] key + mouse wheel zooms all graphs simultaneously. Press the [**Esc**] key to undo zoom and pan for all graphs.
- **Context menu** – Provides the options to change the appearance of graphs. To display the menu, right-click one of the charts – Sensor Data, Sensor Signal, Status, Position – or its legend.
- The context menu options:
    - Increase series line width
    - Decrease series line width
    - Increase selected series line width
    - Decrease selected series line width
    - Zoom in
    - Zoom out.
  - The context menu options for selected graphs also contains:
    - Line style
    - Scatter shape
    - Color
- **Color** – To change a sensor series color on the Legend pane, either double-click or right-click the sensor and select the Color option from the displayed menu. The "Please choose a color for …" dialog displays. You can change the color for the graph lines of the same sensor(widget) on all the charts of the Graph View and on the Touch Signal Graph of the Widget View simultaneously.

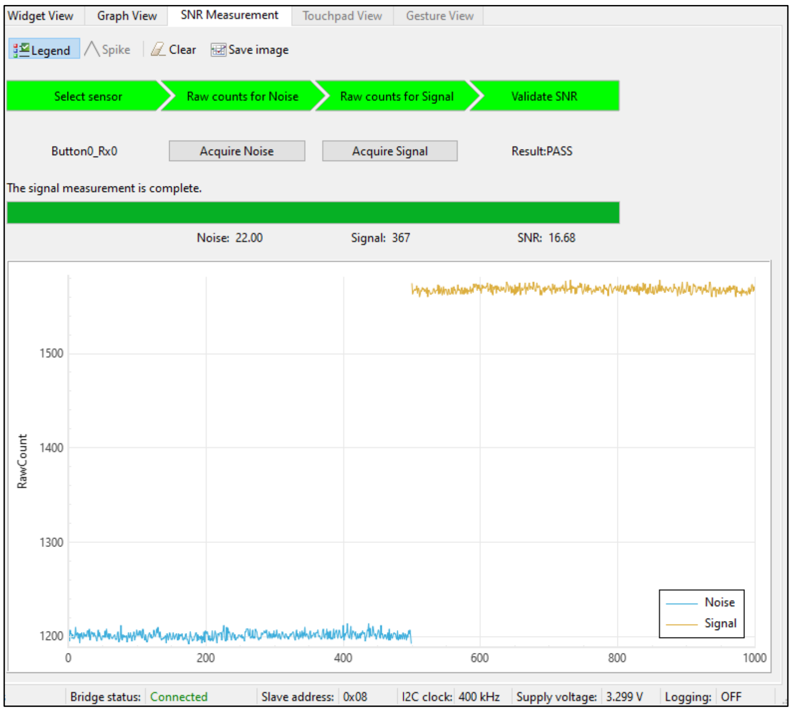


Hue: 224
Red: 12
Sat: 228
Green: 38
Val: 113
Blue: 113
Alpha channel: 255
HTML: #0c2671

Select a color and click **OK**. To select a color for all the graphs at the same time, press and hold the [**Ctrl**] key while clicking.

8 SNR Measurement tab

The SNR Measurement tab allows measuring an SNR for individual sensors. It provides the user interface to acquire noise and signal separately and then calculates an SNR based on the captured data. The obtained value is then validated by a comparison with the required minimum (5 by default, can be configured in the Options dialog).



Button0_Rx0

Result:PASS

The signal measurement is complete.

Noise: 22.00 Signal: 367 SNR: 16.68

Bridge status: Connected Slave address: 0x08 I2C clock: 400 kHz Supply voltage: 3.299 V Logging: OFF

8.1 Toolbar

The toolbar provides quick access to different Tuner configuration options that affect the SNR Measurement graphs display:

- Legend – Shows or hides the legend on the graph.
- Spike – Highlights spikes on the graph.
- Clear – Clears a graph. Note that using this button while noise calculation is in progress will clear all acquired samples and the Tuner will wait for fresh samples to calculate noise.
- Save image – Opens the dialog to save the SNR Measurement tab as an image. To save the whole main window as an image, press and hold the [**Ctrl**] key. The supported formats: .PNG, .JPG, BMP.

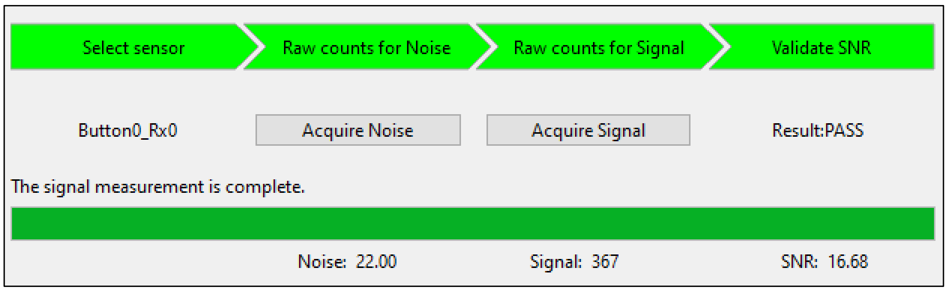
8.2 SNR Measurement procedure

1. Click **Connect** to the device and then **Start** communication on the toolbar.
2. Switch to the SNR Measurement tab.
3. Select the sensor by clicking it on the Widget Explorer pane and observe the **Select sensor** block become green.
4. Ensure no touch is present on the selected sensor.
5. Click **Acquire Noise** and wait for the required count of samples to be collected. Observe the **Raw counts for Noise** block become green after the action completes.

6. Observe the Noise parameter be updated with the calculated noise average value.
7. Touch the selected sensor to produce a signal on it.
8. Click **Acquire Signal** and wait for the required count of samples to be collected. Observe the **Raw counts for Signal** block become green after the action completes.
9. Observe the Signal parameter be updated with the calculated signal average value.
10. Observe the SNR parameter be updated with the SNR. The **Validate SNR** block becomes green if the result is PASS, if FAIL – red.

8.3 SNR measurement pane

The SNR Measurement procedure pane indicates the SNR measurement stages. The status is defined by the background color: green means the stage is successfully completed.



The signal measurement is complete.

Noise: 22.00 Signal: 367 SNR: 16.68

The progress bar provides graphical guidance to the user to complete the SNR measurement. The bar displays the progress and user-addressed relevant messages.

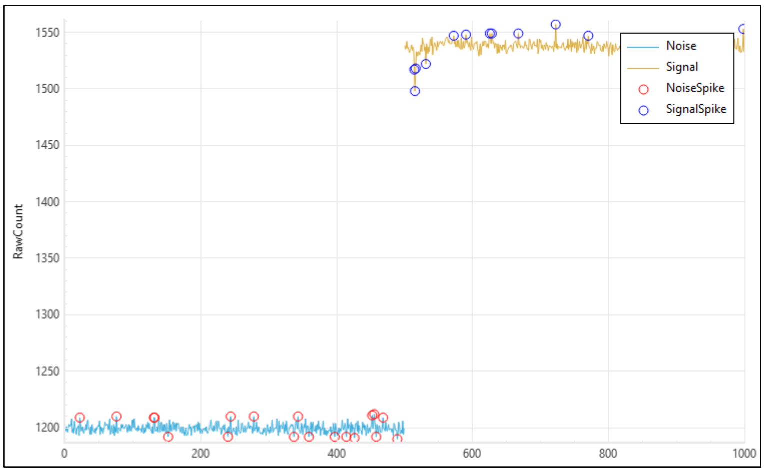
The measured noise, signal, and SNR values are displayed below the progress bar. The SNR value is calculated as Signal divided by Noise rounded up to 2 decimal points.

The result has the following meaning:

a. PASS – The SNR is above the required limit (green).
b. FAIL – The SNR is below the required limit (red).
c. N/A – The SNR cannot be calculated because noise/signal samples are not collected yet (grayed out).

8.4 Graph chart

The graph chart displays the measured sensor noise and signal. Noise and signal spikes displayed on the chart are the points ignored during the SNR calculation.



Click the **Legend** button to make it visible or invisible.

- **Context menu** – Provides the options to change the appearance of graphs. To display the Context menu, right-click any chart.

  The options:

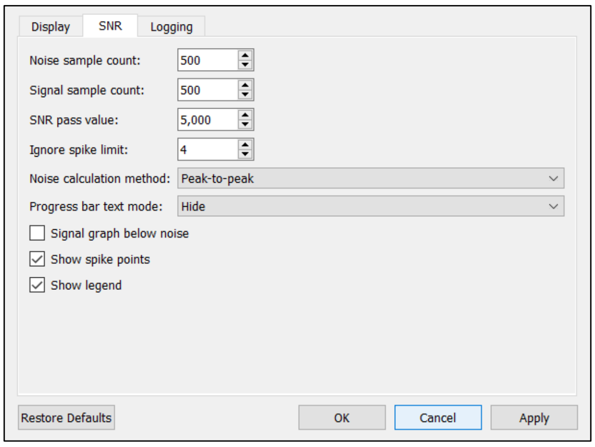
  - Increase series line width
  - Decrease series line width
  - Increase selected series line width
  - Decrease selected series line width
  - Position of Legend (six directions)
  - Legend
  - Spike
  - Clear
  - Save image

  The context menu options for selected graphs also contains:

  - Line style (for Noise and Signal only)
  - Scatter shape
  - Set color for …

8.5 SNR options

SNR parameters can be modified in the options dialog. See Tuner Configuration Options for descriptions of other tabs on this dialog.



Noise sample count: 500
Signal sample count: 500
SNR pass value: 5,000
Ignore spike limit: 4
Noise calculation method: Peak-to-peak
Progress bar text mode: Hide
☐ Signal graph below noise
☑ Show spike points
☑ Show legend

- Noise sample count – The count of samples to acquire during the noise measurement operation.
- Signal sample count – The count of samples to acquire during the signal measurement operation.
- SNR pass value – The minimal acceptable value of the SNR.
- Ignore spike limit – Ignores a specified number of the highest and the lowest spikes at noise / signal calculation. That is, if you specify number 3, then three upper and three lower raw counts are ignored separately for the noise calculation and for the signal calculation.
- Noise calculation method – Allows selecting the method to calculate the noise average. The following methods are available for selection:
  - Peak-to-peak (by default) – Calculates noise as a difference between the maximum and minimum value collected during the noise measurement.
  - RMS – Calculates noise as a root mean-square of all samples collected during the noise measurement.
- Progress bar text mode – This label is shown with the progress bar:
  - Hide (by default) – No label.
  - Percent – The number of samples in percent acquired during the signal measurement operation.
  - Value – The number of samples acquired during the signal measurement operation.

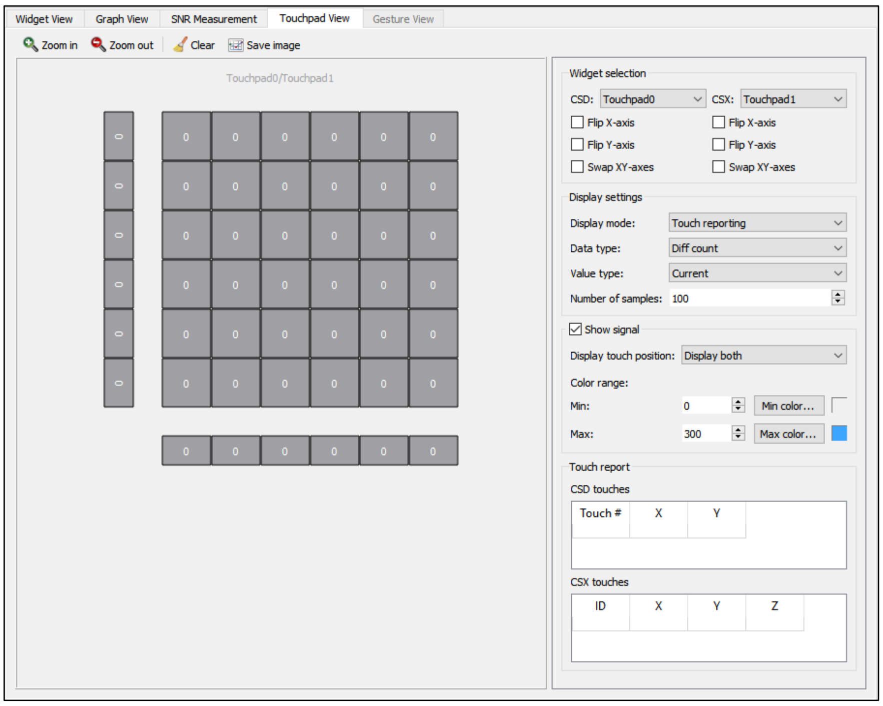
Note: This option is not available on Mac OS.

- Signal graph below noise – Displays the signal series below the noise threshold on the graph.
- Show spike points – Highlights the spike points on the graph.
- Show legend – Shows the graph legend. The Legend has the context menu to change its position. Right-click the Legend to see the menu.

9 Touchpad View tab

This tab visually represents signals and positions of a selected touchpad widget in the heatmap form. Only one CSD and one CSX touchpad can be displayed at a time.

Note: The Touchpad View tab is disabled when there are no touchpad widgets in the configuration.



CSD: Touchpad0 CSX: Touchpad1
☐ Flip X-axis ☐ Flip X-axis
☐ Flip Y-axis ☐ Flip Y-axis
☐ Swap XY-axes ☐ Swap XY-axes

Display mode: Touch reporting
Data type: Diff count
Value type: Current
Number of samples: 100
☑ Show signal
Display touch position: Display both
Color range:
Min: 0
Max: 300

CSD touches

Touch #	X	Y

CSX touches

ID	X	Y	Z

9.1 Toolbar

- Zoom in – Zooms in the Touchpad widget. Press the [**Esc**] key to undo zoom.
- Zoom out – Zooms out the Touchpad widget. Press the [**Esc**] key to undo zoom.
- Clear – Clears the touchpad.
- Save image – Opens a dialog to save the Touchpad View tab as an image. To save the whole main window as an image, press and hold the [**Ctrl**] key. The supported formats The supported formats: .PNG, .JPG, .BMP.

9.2 Widget selection

Consists of the configuration options for mapping the physical touchpad orientation to the identical representation in the heatmap:

- CSD combo box – Selects any CSD touchpad to display in the heatmap.
- CSX combo box – Selects any CSX touchpad to display in the heatmap.
- Flip X-axis – Flips the displayed X-axis to match the orientation of a physical touchpad.
- Flip Y-axis – Flips the displayed Y-axis to match the orientation of a physical touchpad.
- Swap XY-axes – Swaps the X- and Y-axes for the touchpad.

9.3 Display settings

Manages heatmap data to display. These options are applicable for the CSX touchpad only.

- Display mode – The drop-down menu with 3 options for the display format:
  - Touch reporting – Shows current detected touches only.
  - Line drawing – Joins the previous and current touches in a continuous line.
  - Touch Traces – Plots all the reported touches as dots.
- Data type – The drop-down menu to select the signal type to display: Diff count, RawCount, and Baseline. Under Multi-frequency mode, RawCount and Baseline are available for three channels: 0, 1, 2.
- Value type – The drop-down menu to select the type of a value to display:
  - Current – The last received value.
  - Max hold – The maximum value out of latest “Number of samples” received.
  - Min hold – The minimum value out of latest “Number of samples” received.
  - Max-Min – The difference between maximum and minimum values.
  - Average – The average value of latest “Number of samples” received.
- Number of samples – Defines a length of history of data for the Line Drawing, Touch Traces, Max hold, Min hold, Max-Min, and Average options.

9.4 Show signal

Enables displaying data for each sensor if checked, otherwise displays only touches. This option is applicable for the CSX touchpad only.

- Display touch position – Selects the touchpad from which data is displayed in the heatmap. The three options:
  - Display only CSX
  - Display only CSD
  - Display both
- Color range – Defines a range of sensor signals within which the color gradient is applied. If a sensor signal is outside the range, a sensor color is either minimum or maximum out of the available color palette.

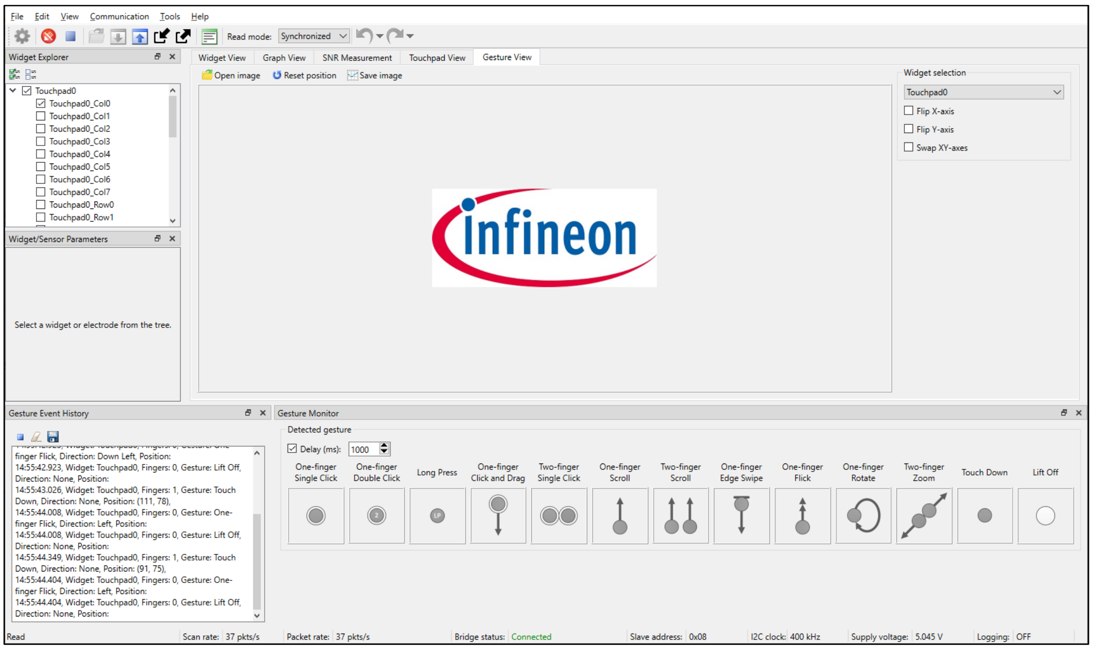
9.5 Touch report

- CSD touches table – Displays the current X and Y touch position (Z value is always 0) of the detected touches on the CSD touchpad. If the CSD touchpad is neither configured nor touch is detected, the touch table is empty. When two-finger detection is enabled for a CSD touchpad, then two touch positions are reported.
- CSX touches table – Displays the current X and Y touch position and Z values (amplitude) of the detected touches on the CSX touchpad. If the CSX touchpad is neither configured nor touches is detected, the touch table is empty. The middleware supports simultaneous detection up to three touches for a CSX touchpad touch, so the touch table displays all the detected touches.

10 Gesture View tab

This tab visually represents evaluation and tuning of gestures (from one widget at a time).

Note: *The Gesture View tab is disabled when there are no gesture widgets in the configuration.*



Select a widget or electrode from the tree.

```
finger Flick, Direction: Down Left, Position:
14:55:42.923, Widget: Touchpad0, Fingers: 0, Gesture: Lift Off,
Direction: None, Position:
14:55:43.026, Widget: Touchpad0, Fingers: 1, Gesture: Touch
Down, Direction: None, Position: (111, 78),
14:55:44.008, Widget: Touchpad0, Fingers: 0, Gesture: One-
finger Flick, Direction: Left, Position:
14:55:44.008, Widget: Touchpad0, Fingers: 0, Gesture: Lift Off,
Direction: None, Position:
14:55:44.349, Widget: Touchpad0, Fingers: 1, Gesture: Touch
Down, Direction: None, Position: (91, 75),
14:55:44.404, Widget: Touchpad0, Fingers: 0, Gesture: One-
finger Flick, Direction: Left, Position:
14:55:44.404, Widget: Touchpad0, Fingers: 0, Gesture: Lift Off,
Direction: None, Position:
```

Read | Scan rate: 37 pkts/s | Packet rate: 37 pkts/s | Bridge status: Connected | Slave address: 0x08 | I2C clock: 400 kHz | Supply voltage: 5.045 V | Logging: OFF

Note: *Synchronized communication mode or UART communication is recommended for Gesture validation, to make sure no gesture event such as a touchdown or lift-off is missed during communication.*

10.1 Toolbar

- Open image – Opens a custom image.
- Reset position – Resets the image position and zoom. The image is moved to the center of the panel.
- Save image – Opens a dialog to save the Gesture View tab as an image. To save the whole main window as an image, press and hold the [**Ctrl**] key. The supported formats: .PNG, .JPG, .BMP.

10.2 Widget selection

Allows selecting a widget and controls that the display in the Tuner matches the orientation physical widget on hardware.

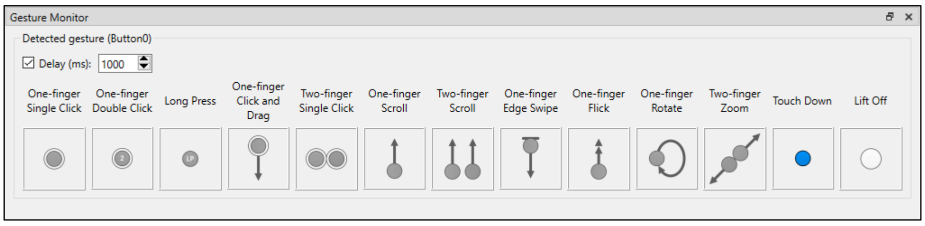
- Combo box – Selects the widget with Gesture enabled to display the Gesture from it on this pane.
- Flip X-axis – Flips the direction of the X-axis to match the orientation of a physical widget.
- Flip Y-axis – Flips the direction of the Y-axis to match orientation of a physical widget.
- Swap XY-axes – Swaps the X- and Y-axes to match orientation of a physical widget.

10.3 Image pane

An image reacts to the detected gestures (scroll and zoom) and moves around the pane. You can change the image by clicking the **Open image** tool button.

10.4 Gesture Monitor

Provides visual indication for a detected gesture. It shows gestures for one widget at a time. The widget can be chosen in the Gesture View tab.



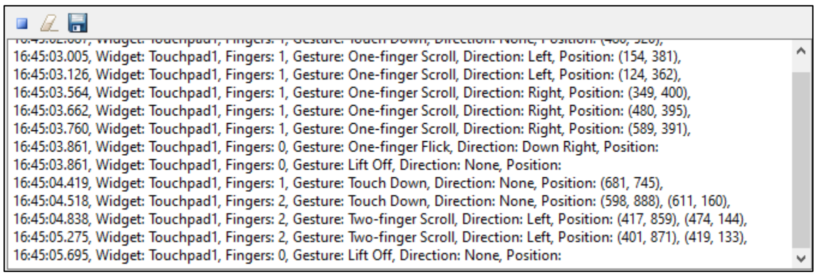
Detected gesture (Button0)

☑ Delay (ms): 1000

If the Delay checkbox is enabled, a Gesture picture is displayed only for the specified time-interval. If disabled, the last reported gesture picture is displayed until a new Gesture is reported.

10.5 Gesture Event history

Logs the detected gestures information.

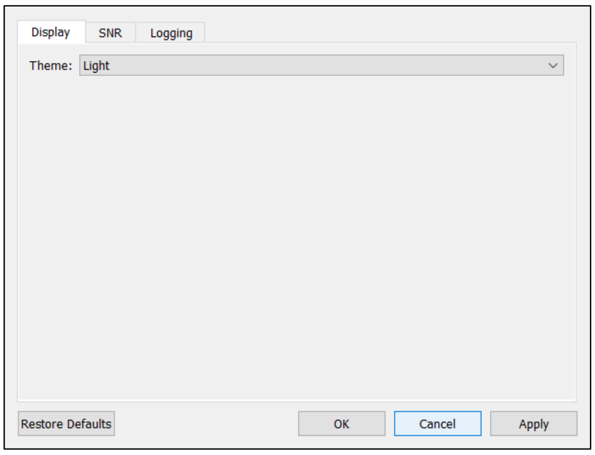


```
16:45:03.005, Widget: Touchpad1, Fingers: 1, Gesture: One-finger Scroll, Direction: Left, Position: (154, 381),
16:45:03.126, Widget: Touchpad1, Fingers: 1, Gesture: One-finger Scroll, Direction: Left, Position: (124, 362),
16:45:03.564, Widget: Touchpad1, Fingers: 1, Gesture: One-finger Scroll, Direction: Right, Position: (349, 400),
16:45:03.662, Widget: Touchpad1, Fingers: 1, Gesture: One-finger Scroll, Direction: Right, Position: (480, 395),
16:45:03.760, Widget: Touchpad1, Fingers: 1, Gesture: One-finger Scroll, Direction: Right, Position: (589, 391),
16:45:03.861, Widget: Touchpad1, Fingers: 0, Gesture: One-finger Flick, Direction: Down Right, Position:
16:45:03.861, Widget: Touchpad1, Fingers: 0, Gesture: Lift Off, Direction: None, Position:
16:45:04.419, Widget: Touchpad1, Fingers: 1, Gesture: Touch Down, Direction: None, Position: (681, 745),
16:45:04.518, Widget: Touchpad1, Fingers: 2, Gesture: Touch Down, Direction: None, Position: (598, 888), (611, 160),
16:45:04.838, Widget: Touchpad1, Fingers: 2, Gesture: Two-finger Scroll, Direction: Left, Position: (417, 859), (474, 144),
16:45:05.275, Widget: Touchpad1, Fingers: 2, Gesture: Two-finger Scroll, Direction: Left, Position: (401, 871), (419, 133),
16:45:05.695, Widget: Touchpad1, Fingers: 0, Gesture: Lift Off, Direction: None, Position:
```

11 Tuner Configuration options

The Tuner application allows setting different configuration options with the options dialog. The settings are divided into groups.

11.1 Display options

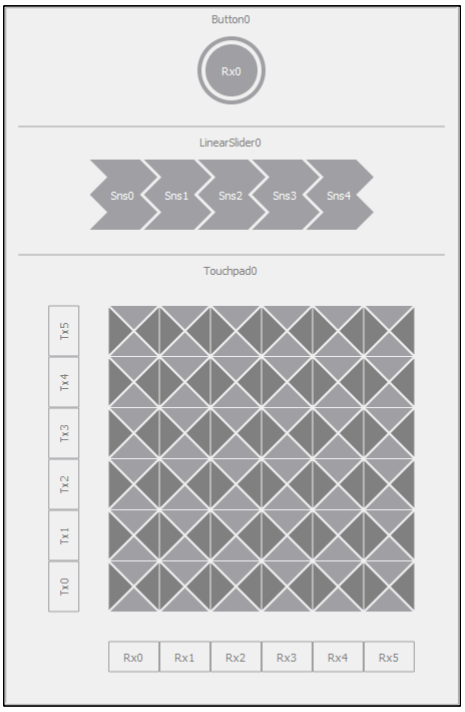


Theme: Light

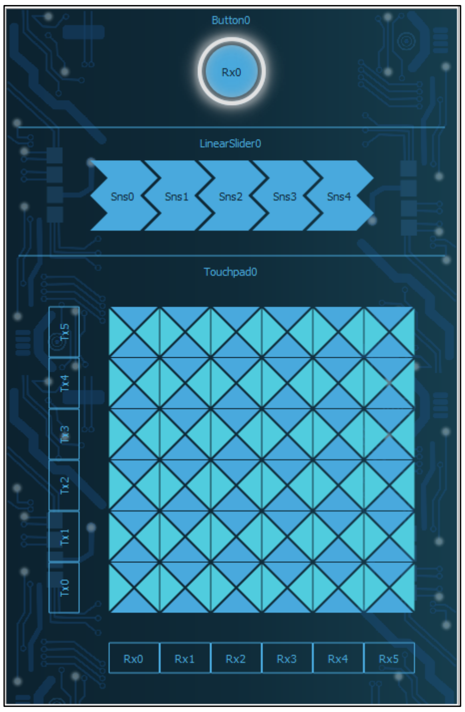
11.1.1 Theme

Defines the visual style of widgets.

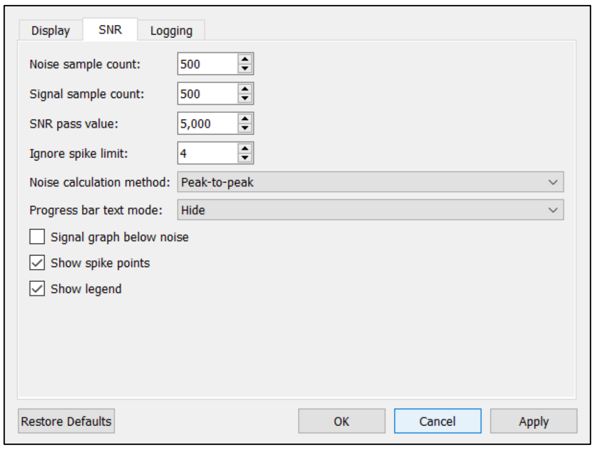
Light theme



Dark theme



11.2 SNR options



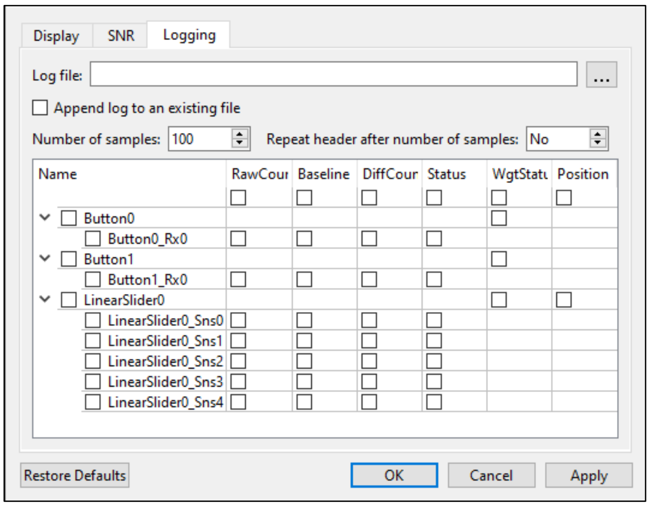
Display | SNR | Logging

Noise sample count:	500
Signal sample count:	500
SNR pass value:	5,000
Ignore spike limit:	4
Noise calculation method:	Peak-to-peak
Progress bar text mode:	Hide

☐ Signal graph below noise
☑ Show spike points
☑ Show legend

See the SNR options section for a description of this tab.

11.3 Logging options



Display | SNR | Logging

Log file:

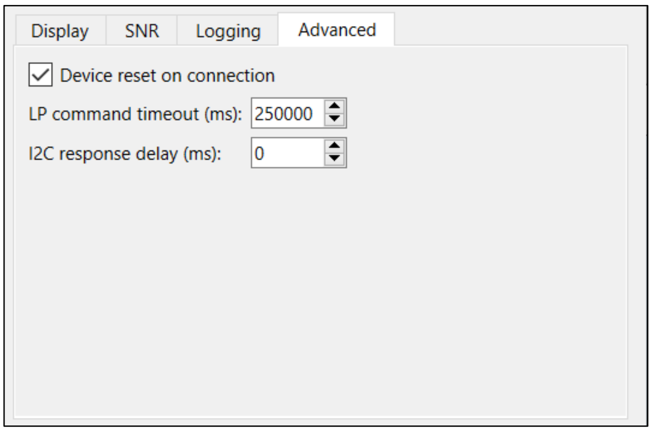
☐ Append log to an existing file

Number of samples: 100   Repeat header after number of samples: No

Name	RawCour	Baseline	DiffCour	Status	WgtStatu	Position
	☐	☐	☐	☐	☐	☐
☐ Button0					☐	
☐ Button0_Rx0	☐	☐	☐	☐		
☐ Button1					☐	
☐ Button1_Rx0	☐	☐	☐	☐		
☐ LinearSlider0					☐	☐
☐ LinearSlider0_Sns0	☐	☐	☐	☐		
☐ LinearSlider0_Sns1	☐	☐	☐	☐		
☐ LinearSlider0_Sns2	☐	☐	☐	☐		
☐ LinearSlider0_Sns3	☐	☐	☐	☐		
☐ LinearSlider0_Sns4	☐	☐	☐	☐		

- Log file – Selects a csv file and its location to store information.
- Append log to an existing file – If this option is enabled, the selected file cannot be overwritten and expanded with new data. When this option is not enabled, the selected file can be overwritten.
- Number of samples – Defines a log session duration in samples. The header is repeated after a number of samples.
- Data configuration check box table – Selects data to collect into a log file.

11.4 Advanced options



[✓] Device reset on connection

LP command timeout (ms): 250000

I2C response delay (ms): 0

- Device reset on connection – If this option is enabled, the target reset is performed on connect and disconnect for 5th generation LP CAPSENSE™ devices which helps to establish connection even if the device was in the deep sleep mode.
- LP command timeout (ms) – Defines the timeout for read and write commands for 5th generation LP CAPSENSE™ devices. During this time the Tuner will repeatedly try to read / write data until the device wakes up from the deep sleep and operation succeeds. Default: 250000 ms.
- I2C response delay (ms) – Configures the delay after the ONE_SCAN command is sent and before the device response is read via the I2C communication protocol in Synchronous mode. The value indicates that the host will not make read requests until the specified time has elapsed. Recommended for long scans to prevent possible noise.

12 Troubleshooting

Problem	**Workaround**
On common Linux distributions, the serial UART ports (usually /dev/ttySx or /dev/ttyUSBx devices) belongs to the root user and to the dialout group. Standard users are not allowed to access these devices.	An easy way to allow the current user access to the Linux machine's serial ports is by adding the user to the dialout group. This can be done using the following command: `$sudo usermod -a -G dialout $USER` *Note: For this command to take effect, you must log out and then log back in.*
On macOS, the CAPSENSE™ Tuner can't be launched from the Launchpad.	Launch the Tuner from within the Device Configurator or use the command line option.
On common Linux distributions, the Tuner forbids communication interface selection after re-plugging KitProg during communication.	Refer to the "Installation Procedure on Ubuntu Linux (x64)" section in the *Cypress Programmer 2.1 CLI User Guide*.
KitProg3 UART is accessible but not able to read data on Linux kernel 4.15 and above or Mac OS X 10.13 and above.	Use third-party UART to USB bridge. Update the KitProg3 firmware to version 1.11.243 or above.
Cannot establish communication with a device that operates in Low power mode.	Refer to the Communication and Low power mode section.
Unable to connect to the device via the RTT communication protocol. The OpenOCD log displays an error "Could not find symbol '_SEGGER_RTT' in executable".	RTT support is not enabled in the application. Refer to the Setting up RTT communication in the application section.

Note: The Tuner provides information about the root cause of connection to KitProg issue in an operation log. The operation log location can be found in the About CAPSENSE™ Tuner box.

13 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 UART interface.
	Added possibility to save different views as images.
	Added zoom and pan functionality for graphs.
	Fixed minor issues.
2.0	Added "IDAC gain index" parameter. Changed the data storage location from a header (.h) file to an XML-based *.cycapsense file. For backward compatibility, the configurator is still able to load the header (.h) file that contains the legacy format configuration. But, if the legacy header (.h) with the configuration is passed via a command-line parameter, a message appears saying that the .h file is not supported. Added the Import and Export options to the File menu that enable importing and exporting the configuration file from and into the external file. Added the Reset View command to the View menu that resets the view to the default. Added Multi-frequency support that enables selecting RawCount and Baseline. Added the UART option description in Status Bar. Changed generation of the middleware initialization structure according to the changes in CapSense v2.0 middleware (related to added fields for flash memory optimization, fixed defect with RawCount filters config, IDAC gain index, etc.). Added support for selecting multiple sensors in some view and reflecting the selection in other views. Added handling of invalid command line arguments. Added highlighting of modified properties in the property grid with bold. Added a warning if opening a broken configuration file. Fixed memory leaks. Fixed Graph View issues under high load of application.
3.0	Added the Undo / Redo feature.
	Changed the list of UART baud-rate possible values to match the values supported by KitProg. The highest rate is 3000000.
3.10	Updated versioning to support patches.
	Added a popup notification when the Apply to device command completes.
	Increased the communication speed between the device and Tuner when using the I2C interface.
	Improved the tool performance for loading large configurations.
	Fixed the issue with applying changes to the device when using the UART communication interface: now, many changes (>16) are applied correctly.
3.11	Updated versioning to support the updated backend, for detail, see Device Configurator user guide.
3.15	Removed from the GUI the possibility to change several UART settings (data bits, stop bits, parity) in order to match KitProg-supported values. The values used: data bits = 8, stop bits = 1, parity = none.
	Optimized the I2C interface communication to increase the speed.
4.0	Added support for the PSoC 4100S Max family.
	Added support for the CAPSENSE™ Middleware Library 3.0.
	Added a CSX touchpad sensor status display on the Graph view.
	Logging data from the device: removed the header duplication from the bottom of the pages of the file.

Version	Change descriptions
	Changed the operation log file location. The new location can be found in the About CapSense Tuner dialog.
5.0	Added support for the 5th generation LP CAPSENSE™ devices.
	Added support for the CAPSENSE™ Middleware Library 4.0.
	Changed the device library file from xml to *props.json*.
	Added the possibility to use the Tuner with USB UART bridges from other vendors.
	Added a set of auxiliary chart commands on the Graph and SNR views.
	Added a panel to display widget and sensor parameters description.
6.0	Added a ‘Scan rate’ field in the status bar. The ‘Refresh rate’ field renamed to ‘Packet rate’.
	Added support of UART communication interface for the 5th generation LP devices.
	Fixed minor display issues.
	Updated information logged in the .log file.
	Changed the structure of the packets sent from the device to the Tuner for the 5th generation LP devices.
6.10	Added support for RTT communication interface.
	Fixed the issue with the displaying button gestures.
	Improved I2C communication speed.
	Added the I2C response delay parameter on the Options tab.

Revision history

Revision	Date	Description
**	2018-11-26	New doc.
*A	2018-12-05	Documents were updated with changes requested in BVI-353.
*B	2019-02-26	Updated to version 1.1.
*C	2019-10-16	Updated to version 2.0.
*D	2019-11-01	Added section for launching from Device Configurator.
*E	2020-03-27	Updated to version 3.0.
*F	2020-09-01	Updated to version 3.10.
*G	2020-12-14	Updated to version 3.11.
*H	2021-03-11	Updated to version 3.15.
*I	2021-09-29	Updated to version 4.0.
*J	2022-09-29	Updated to version 5.0
*K	2023-02-07	Updated to version 6.0
*L	2023-06-01	Updated to version 6.10.

Edition 2023-06-01

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

002-24353 Rev. *L