ModusToolbox™ CAPSENSE™ Tuner user guide

ModusToolbox™ CAPSENSE™ and Multi-Sense Pack
CAPSENSE™ Tuner version 12.0

About this document

A newer revision of this document may be available on the web here.

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.
`Monospace`	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
- ISX – inductive 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
- SMARTSENSE – Auto-tuning process used for proper functionality and optimized performance of the CAPSENSE™ system.
- 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, Touchpad, Low Power, and Liquid Level widget.

About this document

Reference documents

Refer to the following documents for more information as needed:

- ModusToolbox™ tools package user guide
- Eclipse IDE for ModusToolbox™ user guide
- VS Code for ModusToolbox™ user guide
- CAPSENSE™ Configurator user guide
- Device Configurator user guide
- MTB CAT1 Peripheral driver library
- MTB CAT2 Peripheral driver library
- Liquid-level sensing with PSOC™ 4 CAPSENSE™
- Device datasheets
- Device technical reference manuals

Table of contents

About this document . . . 1
Table of contents . . . 3
1 Overview . . . 5
2 Launch the CAPSENSE™ Tuner . . . 6
2.1 From the Device Configurator . . . 6
2.2 make command . . . 7
2.3 VS Code and Eclipse . . . 7
2.4 Executable (GUI) . . . 7
2.5 Executable (CLI) . . . 7
2.6 Parallel design . . . 7
3 Quick start . . . 8
4 GUI description . . . 10
4.1 Main toolbar . . . 11
4.2 Tabs toolbars . . . 12
4.3 Widget Explorer pane . . . 13
4.4 Widget/Sensor Parameters pane . . . 14
4.5 Tabs . . . 16
5 Tuner communication setup . . . 17
5.1 I2C . . . 17
5.2 UART . . . 17
5.3 RTT . . . 18
5.4 Communication and Low power mode . . . 21
5.5 Device power control . . . 22
6 Custom Data Settings . . . 23
7 Status bar . . . 26
8 Widget View tab . . . 27
8.1 Touch Signal Graph . . . 27
9 Graph View tab . . . 28
9.1 Graph View tab toolbar . . . 28
9.2 Graph functionality . . . 29
10 SNR Measurement tab . . . 32
10.1 SNR Measurement procedure . . . 32
10.2 SNR measurement pane . . . 32
10.3 Graph chart (SNR) . . . 33
10.4 SNR options . . . 33
11 Liquid Level Measurement tab . . . 35

Table of contents

11.1 Liquid Level Calibration 35
11.2 Multi-level liquid presence detection 39
11.3 Graph chart (LL) 40
11.4 Save/Load logged data 40

12 Touchpad View tab 41
12.1 Widget selection 41
12.2 Display settings 41
12.3 Show signal 42
12.4 Touch report 42
12.5 Hot keys 42

13 Gesture View tab 44
13.1 Widget selection 44
13.2 Image pane 44
13.3 Gesture Monitor 44
13.4 Gesture Event History 45

14 Tuner Configuration options 46
14.1 Display options 46
14.2 SNR 46
14.3 Logging options 47
14.4 Advanced options 48

15 Troubleshooting 49

16 Version changes 50

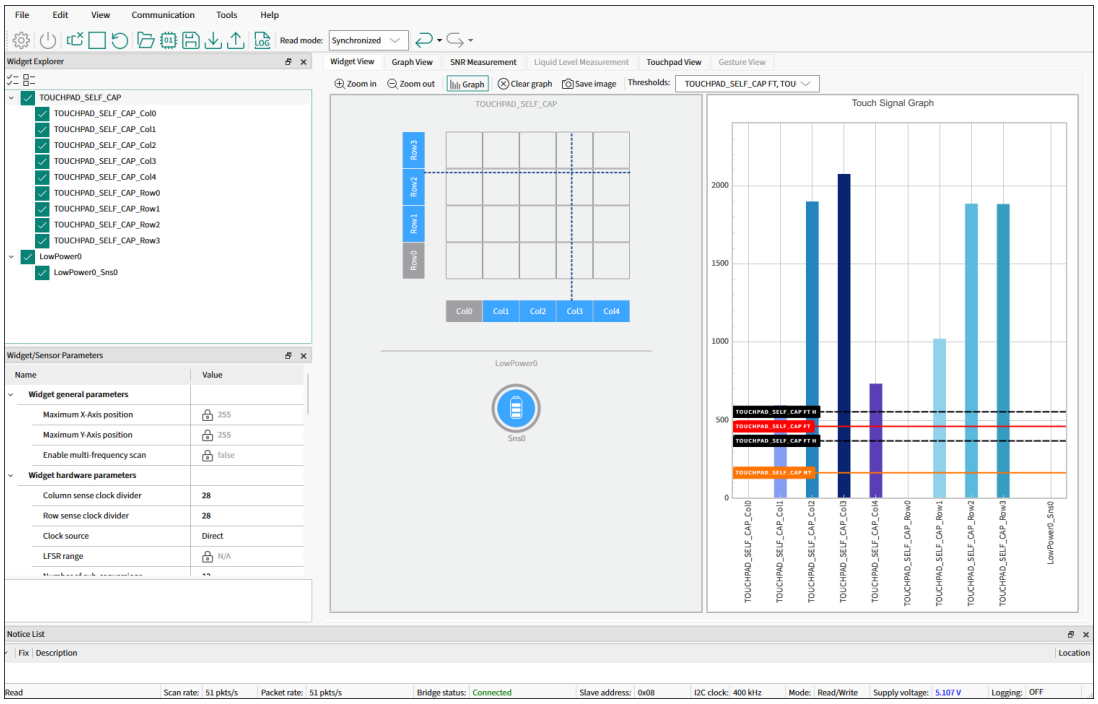
Revision history 53

Disclaimer 54

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 user guide and 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.



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 user guide for more details.

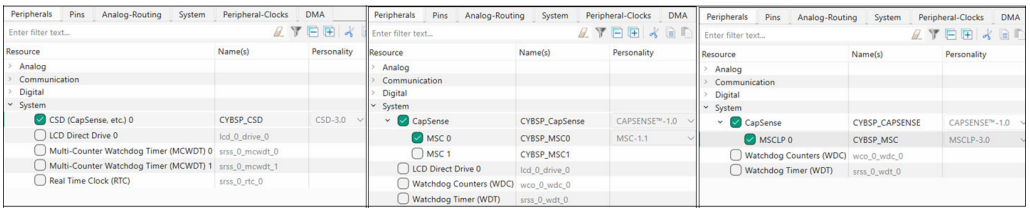
2.1 From the Device Configurator

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

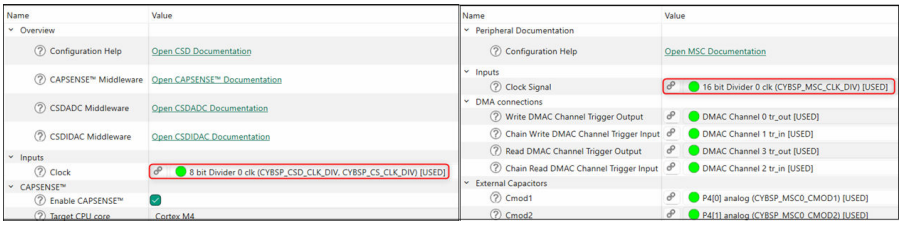
The process differs slightly depending on the hardware block:

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

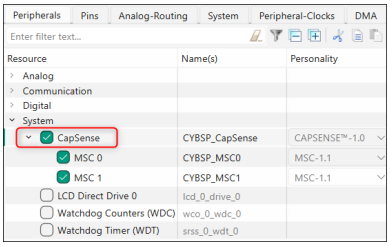
1. On the **Peripherals** tab, select the **CSD (CapSense)** resource or the **CapSense** and one or more **MSC** or **MSCLP** resources, as applicable for your device.



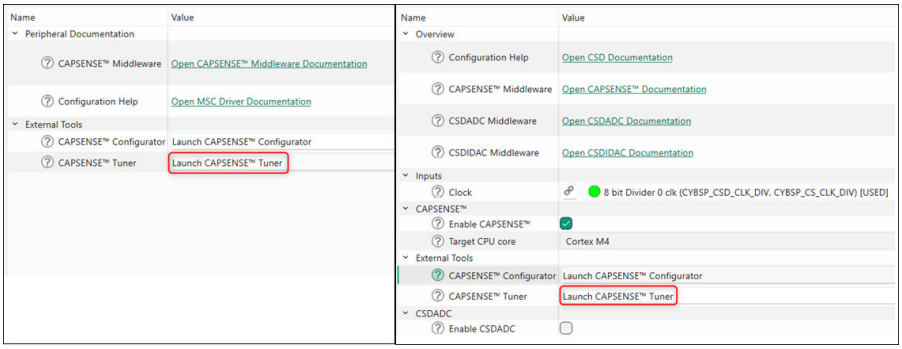
2. (Skip this step for MSCLP.) On the **Parameters** pane, select an appropriate input **Clock**.



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



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



2.2 make command

As described in the ModusToolbox ™ tools package user guide "ModusToolbox 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 VS Code and Eclipse

VS Code and Eclipse have tools to launch the CAPSENSE™ Tuner from within an open application. Refer to the applicable user guide for more details:

- VS Code for ModusToolbox™ user guide
- Eclipse IDE for ModusToolbox™ user guide

2.4 Executable (GUI)

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

<install_dir>/ModusToolbox/packs/ModusToolbox-Multi-Sense-Pack/tools/capsense-configurator

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

2.5 Executable (CLI)

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.

2.6 Parallel design

You can simultaneously open the CAPSENSE™ Tuner and Configurator to work with the same <file_name>.*cycapsense* configuration file. To get the values in the CAPSENSE™ Tuner, which were updated in the CAPSENSE™ Configurator, use the **Import** option or reopen the CAPSENSE™ Tuner.

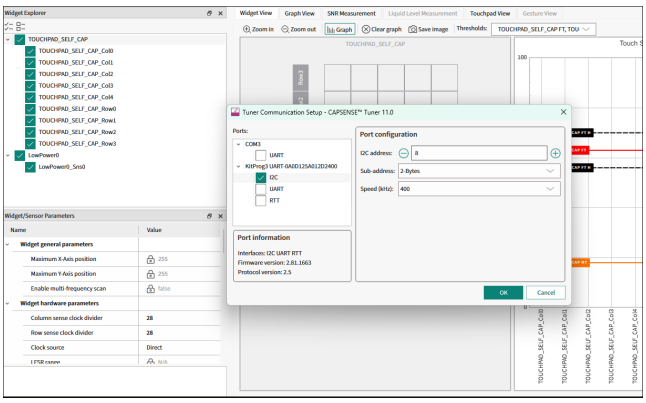
3 Quick start

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 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 configured to work on PSOC™ 4 MCU kits.*

2. Launch the CAPSENSE™ Tuner.
3. Start Communication. Click **Connect** and select the required communication interface.



   ***Note**: For MiniProg, first, power the device – click the **Tuner Communication Setup** button to select the communication interface, use the **Power** button to select and apply the power supply, and proceed with **Connect**.*

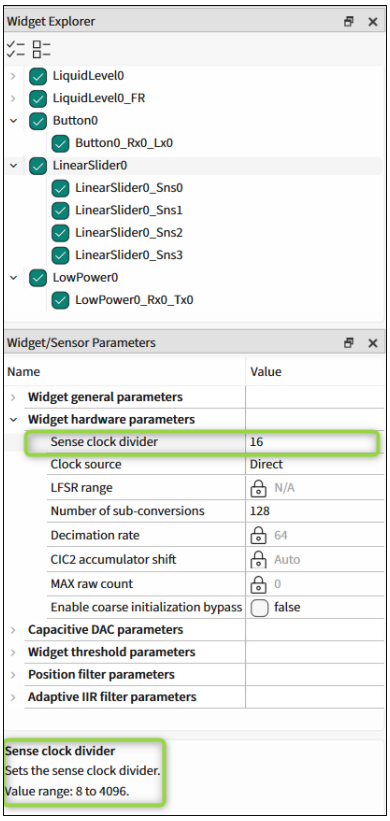
   ***Note**: Refer to the KitProg user guide for the supported configurations and modes. For this example, I2C address, sub-address, and speed must match the configuration.*

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

4. Click **Start** to extract data.

5. Touch the sensors on the hardware and notice the change in the sensor/widget status on Widget View tab.
6. Open the **Graph View** tab. Check the sensors in the Widget Explorer pane to observe sensor signals on the graph. Touch the sensors and notice the signal change on the Graph View tab.
7. Change widget/sensor parameter values as needed. Then, apply the new settings to the device temporarily using the **Apply to Device** command to test these new settings.

3 Quick start



8. Exit the Tuner application. After rebuilding the project and re-programming the device, the changes will be applied to the device permanently.

4 GUI description

The CAPSENSE™ Tuner application contains Menus, Main toolbar and Tabs toolbars, Widget Explorer pane and Widget/Sensor Parameters pane, all used to tune a CAPSENSE™ application.

Menus

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. 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 Read Only mode of the communication interfaces, this command is grayed out.

  ***Note**: These changes are temporary. They are stored in the device operational memory.*

  To save the changes:

  - *apply them to the project*
  - *rebuild the project*
  - *re-program the device*

- **Apply to Project** – Commits currently changed values of widget/sensor parameters to the CAPSENSE™ project.
- **Open is System Explorer** – This opens your computer's File Explorer window 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 Map to PDF** – Exports the current configuration register map in PDF format for the latest version of the middleware available on the date of the CAPSENSE™ Tuner release.
- 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.

Edit

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

View

- **Widget Explorer** – Hides or shows the Widget Explorer pane where widgets and sensors tree display.
- **Notice List** – Hides or shows the Notice List pane. The pane displays by default.
- **Widget/Sensor Parameters** – Hides or shows the Widget/Sensor Parameters 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.
- **OpenOCD Log** – Hides or shows the OpenOCD Log pane.
- **Toolbar** – Hides or shows the toolbar. Enabled by default.

  ***Note**: You can dock the Tuner panes and toolbars anywhere on the interface.*

- **Reset View** – Resets the view to the default.

Communication

- **Power** – This button powers on/off the target device.

- **Connect** – Connects to the device via a communication channel selected in the "Tuner Communication Setup" dialog. If the channel was not previously selected, the dialog displays.
- **Disconnect** – Closes the communication channel with the connected device.
- **Start** – Starts reading data from the device.
- **Stop** – Stops reading data from the device.
- **Restart** – Restarts reading data from the device.

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 the Tuner preferences: **Display**, **SNR**, **Logging**, **Advanced**.
- **Custom Data Settings…** – Opens either from under **Tools** or from the **Custom Data** graph. The name, data type, and byte offset of the variables are defined by the user.
- **Logging** – Starts/stops data logging into a specified file.

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.

Notice List

The **Notice List** pane combines notices (errors, warnings, tasks, and infos) from many places in the configuration into a centralized list. You can double-click a notice location to show the parameter causing the error or warning. For more information about the Notice List, refer to Device Configurator user guide.

4.1 Main toolbar

The main toolbar contains buttons for the commands that duplicate the Menus items.

Also, the main toolbar contains:

- **Read mode** – Selects the Tuner communication mode with a device (I2C and UART with LP).
  - **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.

4.2 Tabs toolbars

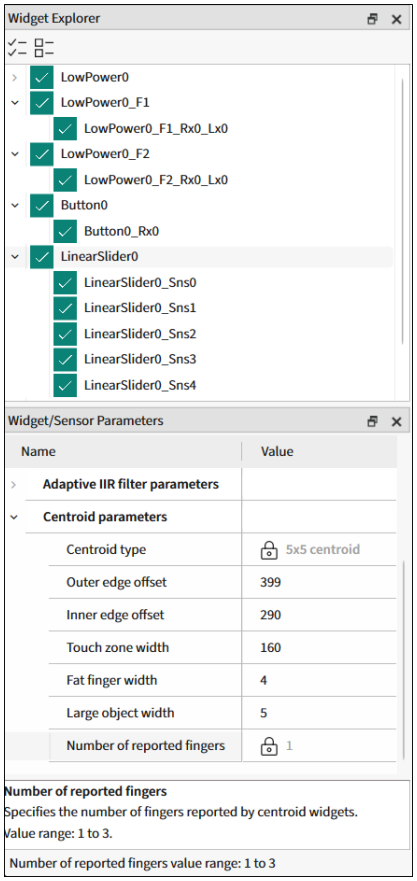
The toolbars options of the CAPSENSE™ Tuner tabs by tab:

Option	Description	Widget View tab	Graph View tab	SNR Measurement tab	Liquid Level Measurement tab	Touchpad View tab	Gesture View tab
Zoom in/ Zoom out (Graph functionality)	Zooms in/out the widgets. Press the [**Esc**] key to undo zoom.	√	√			√	
Graph	Shows or hides the Touch Signal Graph.	√					
Save image	Opens the dialog to save the tab view as an image. To save the whole main window as an image, press and hold the [**Ctrl**] key. The supported formats: .PNG, .JPG, BMP.	√	√	√	√	√	√
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		√	√	√		
Clear graph/ Clear	Clears all graphs.	√	√	√	√	√	
Spike	Highlights spikes on the graph.			√			
Opaque	The level of opacity of the area below the line series.		√				
Chart settings (Graph View tab toolbar)	Opens the menu for editing the chart line series parameters		√				
Reset position	Resets the image position and zoom. The image is moved to the center of the panel.						√
Open image	Opens a custom image.						√

Option	Description	Widget View tab	Graph View tab	SNR Measurement tab	Liquid Level Measurement tab	Touchpad View tab	Gesture View tab
Save logged data	Saves data to a *.csv* file, by default to the *design.lls.csv* file.				√		
Load logged data	Logs data to a *.csv* file, by default to the *design.lls.csv* file.				√		
Thresholds (Graph View tab toolbar)	Displays on the sensor charts with labels that correspond to the widget threshold names selected on the drop-down list.	√	√				

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 tab and the position graph series in the Graph View tab while the sensor checked status controls the visibility of the sensor raw count/baseline/signal/status graph series in the Widget View tab and signals in the Touch Signal Graph on the Widget View tab.



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 ISX widgets, the sensor tree displays individual nodes (Rx0_Lx0, Rx0_Lx1 …) contrary to the CAPSENSE™ Configurator where the ISX electrodes are displayed (Rx0, Rx1 … Lx0, Lx1 ...).*

***Note**: For MSC HW configurations with enabled Multi-frequency mode, the Median check box in the Widget Explorer for each sensor displays a set of F0 values received from a kit in the Graph View tab tab.*

4.4 Widget/Sensor Parameters pane

The **Widget/Sensor Parameters** pane displays the parameters of a widget or sensor selected in theWidget Explorer pane tree. The parameters values display as follows:

- without connection to a device – values are taken from the configuration file
- with connection to a device – values are read directly from the device; for parameters set to Auto, the actual device value displays.

You can modify any editable parameters and apply changes to the device at run-time. Select a parameter to see its description on the panel below the parameters list.

The Widget/Sensor Parameters pane includes groups of parameters described in the Widget parameters and Sensor parameters sections.

4.4.1 Widget parameters

Widget general parameters

Most of these parameters are read-only in the CAPSENSE™ Tuner because corresponding parameter values reside in the application flash widget structures that cannot be modified at run-time.

Widget hardware parameters

These parameters can be tuned for the widgets without SMARTSENSE. Require hardware re-initialization. In Manual tuning mode, the availability of changing the Widget hardware parameters and **Apply to Device** command depends on the mode and communication protocol.

Protocol	**I2C**	**UART**	**RTT**
Mode	Read/Write		
Apply to Device	√	√	√
Widget hardware parameters	√	√	√
Mode	Read only		
Apply to Device	√		
Widget hardware parameters		√	√

Capacitive DAC parameters

These parameters are read-only for the widgets with SMARTSENSE or appropriate CDAC mode set to Auto in the CAPSENSE™ Configurator.

Scaled capacitive DAC parameters

These parameters are:

- available for the 5th generation LP devices
- applicable for the CSD sensing method
- read-only.

Widget threshold parameters

These parameters are read-only for widgets with SMARTSENSE– Full. In Manual tuning mode (for all widgets), the availability of changing Widget threshold parameters and **Apply to Device** command depends on the mode and communication protocol.

Protocol	**I2C**	**UART**	**RTT**
Mode	Read/Write		
Apply to Device	√	√	√
Widget threshold parameters	√	√	√
Mode	Read only		
Apply to Device	√		
Widget threshold parameters	√	√	√
	Note: *Except ON debounce parameter (requires hardware re-initialization).*		

Position filter/Adaptive IIR filter/Ballistic multiplier/Centroid parameters

These parameters cannot be modified at run-time from the Tuner because corresponding parameter values reside in the application flash widget structures that cannot be modified at run-time.

4.4.2 Sensor parameters

Sensing parameters

- When IDAC/CDAC 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 Read/Write mode.

Capacitance (fF)

When BIST is enabled in the CAPSENSE™ Configurator, the sensor/electrode capacitance is read-only, and displays values calculated by the firmware. The application code must have BIST-related function calls. When BIST is disabled, the Capacitance (fF) group is hidden

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.*

Note: *For Liquid Level widgets, there is **Level filter parameters** group instead of **Position filter parameters** group.*

4.5 Tabs

The CAPSENSE™ Tuner consists of the following tabs:

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

5 Tuner communication setup

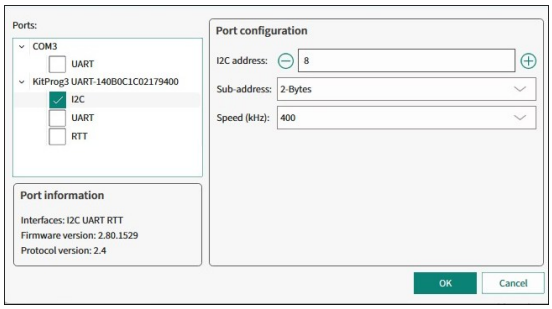
The "Tuner Communication Setup" dialog is used to establish communication between the Tuner and target device. The Tuner supports I2C, UART, and RTT communication interfaces.

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.*

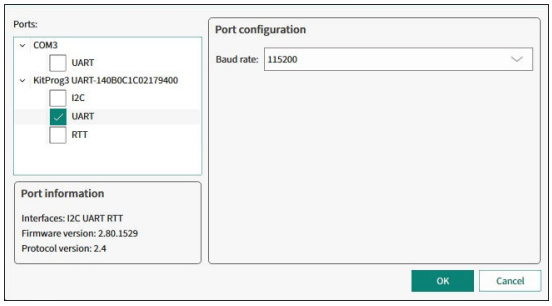
5.1 I2C



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 the two Read modes (see Main toolbar).

5.2 UART



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: Read/Write and Read Only (see Status bar).

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

5.3 RTT

The 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.

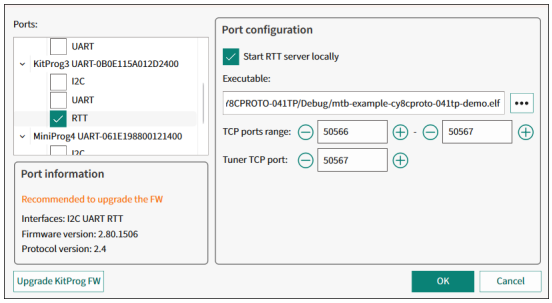
The data transfer between the host and target devices is organized through channels. The Tuner communicates to the server (OpenOCD) via the TCP/IP protocol. The server in turn uses the SWD interface to communicate with the device.

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 Setting up RTT communication in application.

5.3.1 Start RTT server locally



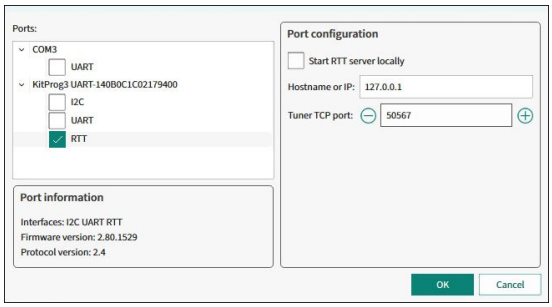
To start the RTT server in your local environment, select **Start RTT server locally** and modify the following fields:

- **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 located with the Tuner executable (or *capsense-tuner.app/Contents/Resources/* on macOS). It starts an OpenOCD session that lasts while the Tuner is connected to the device.

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

5.3.2 Connect to existing RTT server



To connect to an existing RTT server, deselect **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 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.

Add sample code 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. Set the size of: the up buffer to at least sizeof(cy_capsense_tuner) + 5 + 1 (5 bytes for the packet header and trailer, 1 byte for internal RTT purposes); the down buffer to 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;
...
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.

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:

The following 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, it 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.

5.5 Device power control

The Tuner offers a convenient feature to control the device power using the **Power** button, which provides the following functionalities:

- Monitor the target device power
- Turn the power on/off (if supported)
- Control the target device voltage (if supported)

All kits with KitProg devices support the power monitoring and the target voltage displays on the Tuner status bar. However, not all devices support the powering on/off or adjusting the voltage. The power control behavior depends on the kit type as detailed below:

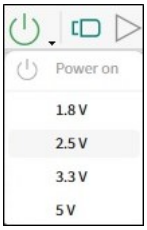
Kits that support power on/off without voltage control

- The **Power** button is enabled but the voltage selection is not available.
- The value of the target voltage is determined by the board design.

***Note**: KitProg does not provide the power supply value before the target device is powered on.*

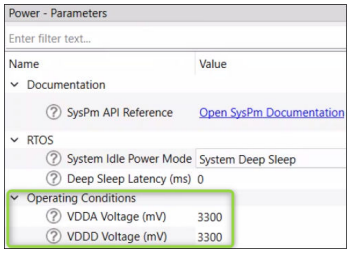
Kits that support both power on/off and voltage control

- To turn the power on, first, select the target voltage value from the available options.



- If the power is already turned on, you can change the voltage by selecting a new value from the drop-down menu.

***Note**: Select the power supply voltage the same as in the Device Configurator under **Operating Conditions**.*



6 Custom Data Settings

The "Custom data Settings" dialog is used to configure, populate, decode, and display application-defined custom data in the CAPSENSE™ Tuner.

Define a packed structure that matches the custom-data buffer layout.

```
typedef __PACKED_STRUCT
{
    uint8_t  var1[10u];
    uint16_t var2;
    uint32_t var3;
} customDataStruct_t;
```

The total required buffer size for the above structure is 16 bytes. In the Configurator, enable the custom-data buffer and edit the custom-data buffer size to the required value. The available buffer size is from 1 to 256 bytes.

6 Custom Data Settings

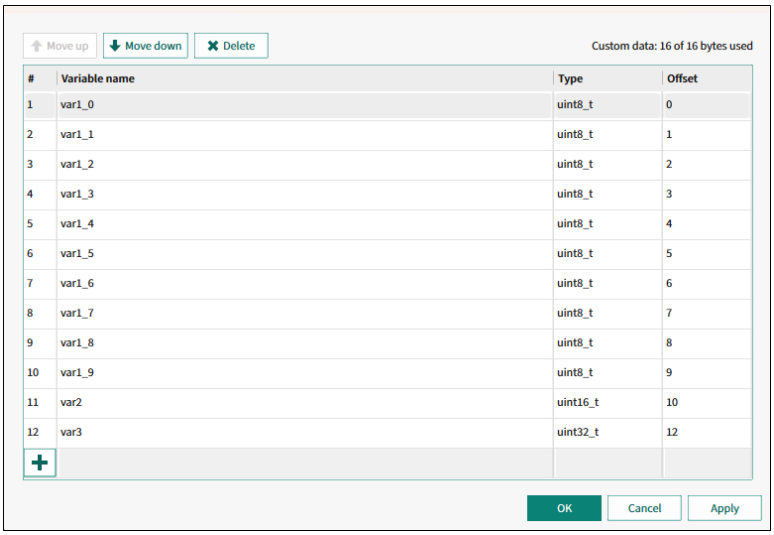
In the application, write custom data before `Cy_CapSense_RunTuner()`.

```
uint8_t i;

/* First scan */
Cy_CapSense_ScanAllWidgets(&cy_capsense_context);

for (;;)
{
    if (CY_CAPSENSE_NOT_BUSY == Cy_CapSense_IsBusy(&cy_capsense_context))
    {
        /* Process raw scan data. */
        Cy_CapSense_ProcessAllWidgets(&cy_capsense_context);

        /* Write application data to the tuner buffer. */
        for (i = 0u; i < sizeof(((customDataStruct_t *)CY_CAPSENSE_CUSTOM_DATA_VALUE)->var1); i+
+)
        {
            ((customDataStruct_t *)CY_CAPSENSE_CUSTOM_DATA_VALUE)->var1[i] =
                cy_capsense_context.ptrWdConfig[0u].ptrSnsContext[0u].status;
        }

        ((customDataStruct_t *)CY_CAPSENSE_CUSTOM_DATA_VALUE)->var2 =
            cy_capsense_context.ptrWdConfig[0u].ptrSnsContext[0u].raw;

        ((customDataStruct_t *)CY_CAPSENSE_CUSTOM_DATA_VALUE)->var3 =
            cy_capsense_context.ptrWdContext[0u].status;

        /* Synchronize the tuner buffer with the tuner. */
        Cy_CapSense_RunTuner(&cy_capsense_context);

        /* Start the next scan. */
        Cy_CapSense_ScanAllWidgets(&cy_capsense_context);
    }
}
```

Configure the custom-data buffer and definitions as follows:

1. In the Tuner, open **Tools > Custom Data Settings...**.
2. Add the variables in the **Custom Data Settings...** dialog and click **OK.**

6 Custom Data Settings



3. Open the Graph View tab.
4. Expand the **Custom Data** chart. Note that it is hidden by default.
5. Select the required variables on the chart legend.

To include custom data in the Tuner log:

1. Open **Tools > Options...**, enable **Custom data** in the Logging options.
2. Select either the byte array or decoded custom-data variables as the log format.
3. Write multi-byte values to the Tuner buffer in the little-endian format for the correct tuner decoding.

The requirements:

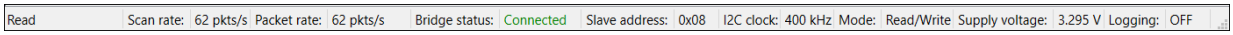
- The configured offsets must match the application structure layout.
- The variable must fit completely inside the configured custom-data buffer.

***Note**: The Tuner uses custom-data definitions only for decoding and visualization/logging. The Tuner does not write values into the application custom-data buffer.*

7 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



UART interface

RTT interface

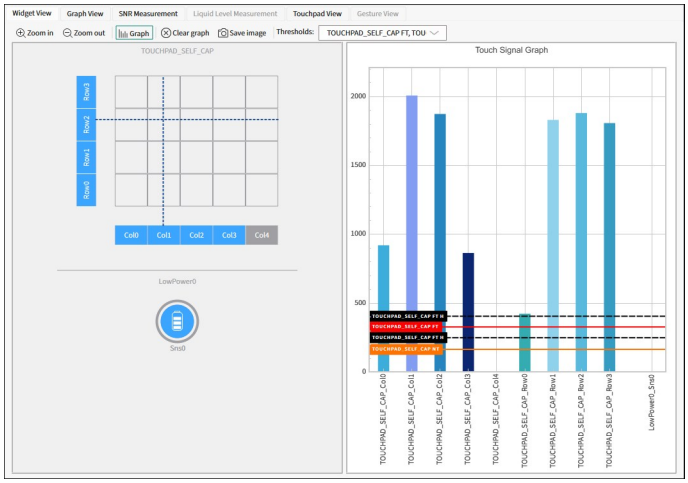
Fields

- **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** – The data transmission mode:
  - Read/Write – Provides data transmission in two reverse directions: target device → Tuner→ target device. You can change the parameters of the widgets/sensors at run-time. To upload the modified configuration, click the **Apply to Device** button.
  - Read only – Provides data transmission in one direction: target device → Tuner. You cannot upload the parameters of the widgets/sensors to the device at run-time. But for the I2C protocol you can still upload parameters that do not require re-initialization.
- **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.

8 Widget View tab

The **Widget View** tab 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 toolbar is described in Tabs toolbars.



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.

8.1 Touch Signal Graph

The **Widget View** tab also displays the Touch Signal Graph. This graph contains a touch signal level for each sensor selected on the Widget Explorer pane.

9 Graph View tab

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

The toolbar is described in Tabs toolbars.

The following graphs are available:

- **Sensor Data** – Check a corresponding check box to display the RawCount and Baseline parameters. In Multi-frequency mode, RawCount and Baseline are available for three channels: 0, 1, 2.
- **Sensor DiffCount** – Displays signal differences.
- **Status** – Displays the sensor status (Touch/No Touch).
- **Position** – Displays touch positions for the Linear Slider, Radial Slider, and Touchpad widgets.
- **Custom Data** – Displays custom application-specific parameters. For more details, refer to Custom Data Settings.

To scale the required parameters, you can either enable the **Auto scale Y** option or select the minimum and maximum scaling values manually in the **Range** combo box.

***Note**: For 5th generation CAPSENSE™ configuration 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.*

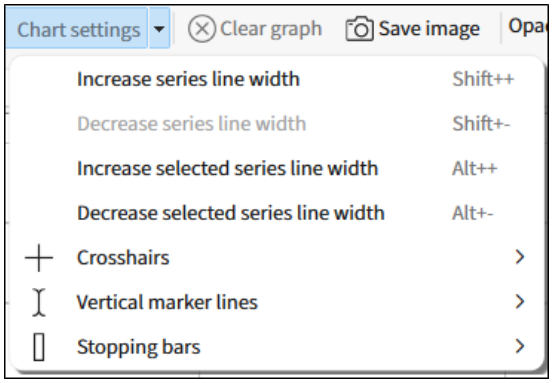
9.1 Graph View tab toolbar

Go to the Tabs toolbars table to see the descriptions of the **Graph View** tab toolbar.

Also, there are **Chart settings** and **Thresholds** drop-down menus on the **Graph View** toolbar.

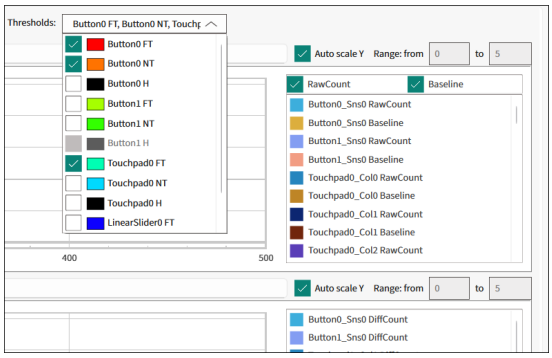
Chart settings

Left -click **Chart settings** to see the menu for editing the chart line series parameters.



- **Crosshairs** – Consists of two lines parallel to the coordinate axes. The vertical component always shows the X coordinate of the mouse cursor and moves with the mouse pointer synchronously on all four charts: **Sensor Data**, **Sensor Signal**, **Status**, and **Position** regardless of their scale and shift. The horizontal component is visible only on the chart (active chart) over which the mouse cursor is hovering. Right-click the **Crosshairs** to see the context menu options.
- **Vertical marker lines** – When you left-click, the Tuner adds a Vertical marker line with a value equal to the X coordinate of the mouse cursor synchronously on all four charts – **Sensor Data**, **Sensor Signal**, **Status**, and **Position** regardless of their scale and shift. Press the [**Esc**] key to erase all previously drawn **Vertical marker lines**. You can change the characteristics of **Vertical marker lines** or disable them in the submenu **Chart settings > Vertical marker lines**.
- **Stopping bars** – When performing **Stop communication**, the Tuner adds a rectangular mark after the current endpoint synchronously on all four charts. After a new start of communication, new chart points will be added after the last stopping bar. You can change the **Stopping bars** characteristics or disable them in the **Chart settings > Stopping bars** submenu.

Thresholds



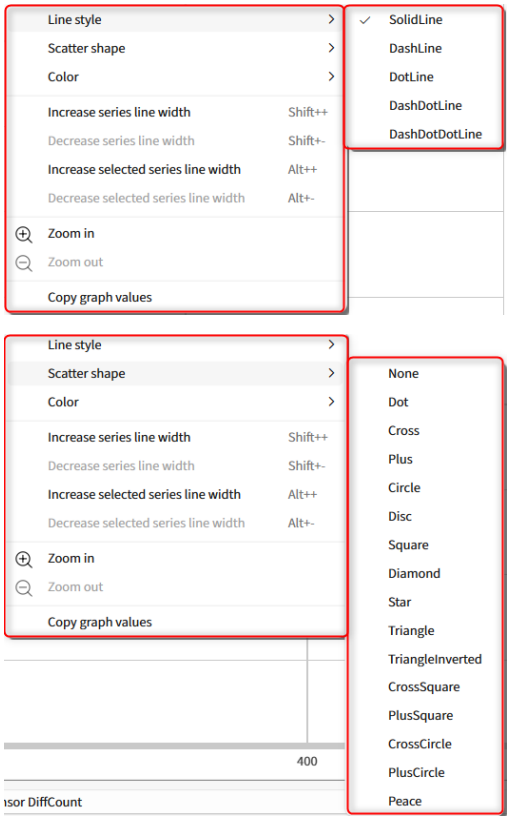
Each threshold line on the sensor DiffCount plot is labeled with the relevant widget and threshold name. Drop-down selections are saved and restored when the Tuner is reopened, all check boxes default to unchecked.

9.2 Graph functionality

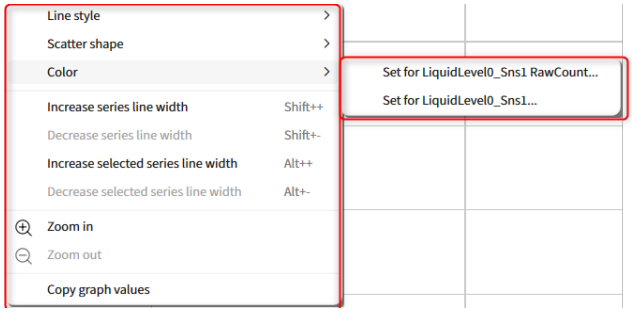
The Graph functionality includes:

- **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
- The 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 the chart's legend.
  - The context menu options are the same as for Graph View tab toolbar**> Chart settings**: **Increase**/ **Decrease series line width** and **Zoom**.
  - The context menu for selected graphs (**Increase**/**Decrease selected series line width**)also contains the **Line style** and **Scatter shape** options:

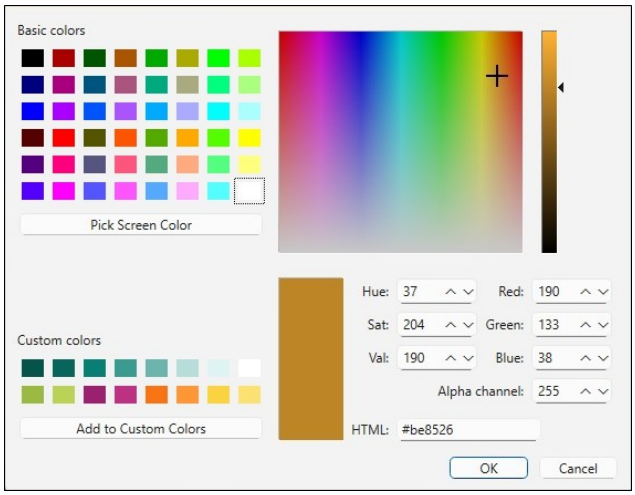


  - To change a sensor series color on the **Legend** pane, either double-click or right-click the sensor and select the **Color** option from the displayed menu. Select the widget for which to set the color.



    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.

9 Graph View tab

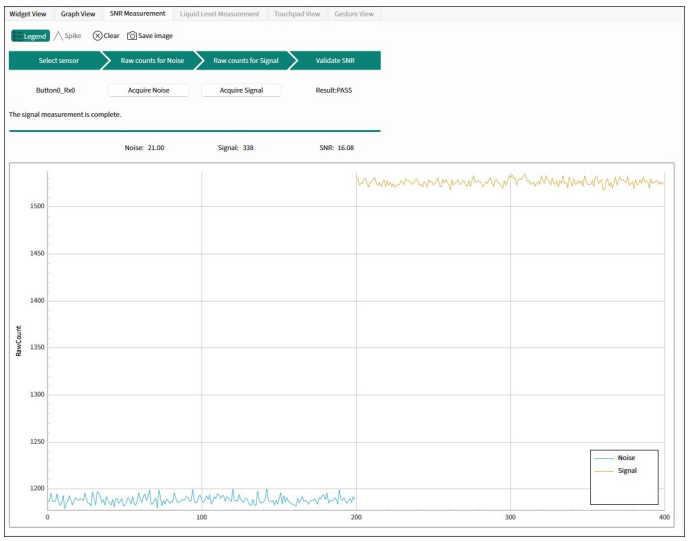


Select a color and click **OK**. To select a color for all the graphs at the same time, press and hold the [**Ctrl**] key while clicking.

10 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).

The toolbar is described in Tabs toolbars.



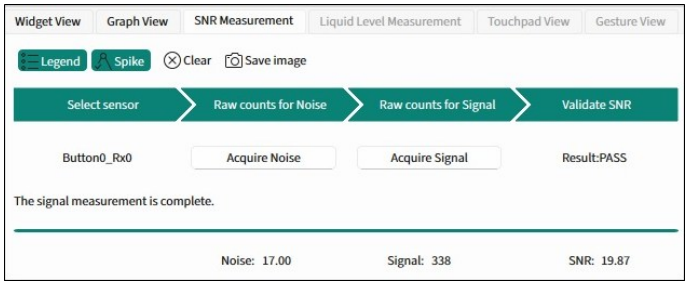
10.1 SNR Measurement procedure

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.

10.2 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.

10 SNR Measurement tab



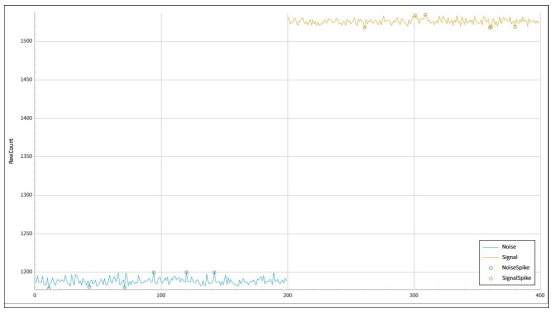
The progress bar provides graphical guidance to the user to complete the SNR measurement. The bar displays the progress and user-addressed relevant messages.

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:

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

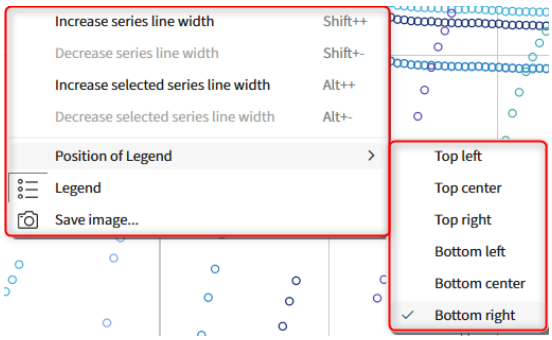
10.3 Graph chart (SNR)

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

Position of Legend (six directions)

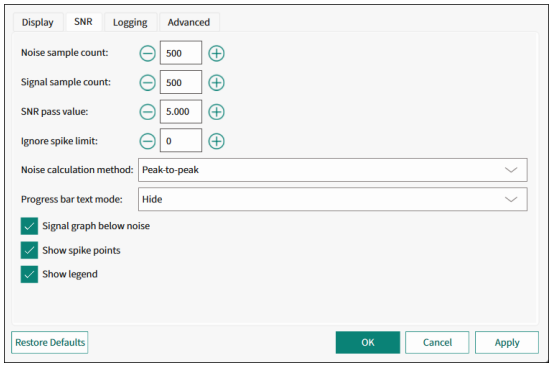


For a description of **Legend**, **Spike**, **Clear**, and **Save image**, go to the Tabs toolbars table.

10.4 SNR options

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

10 SNR Measurement tab



Fields:

- **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 macOS.*

- **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.

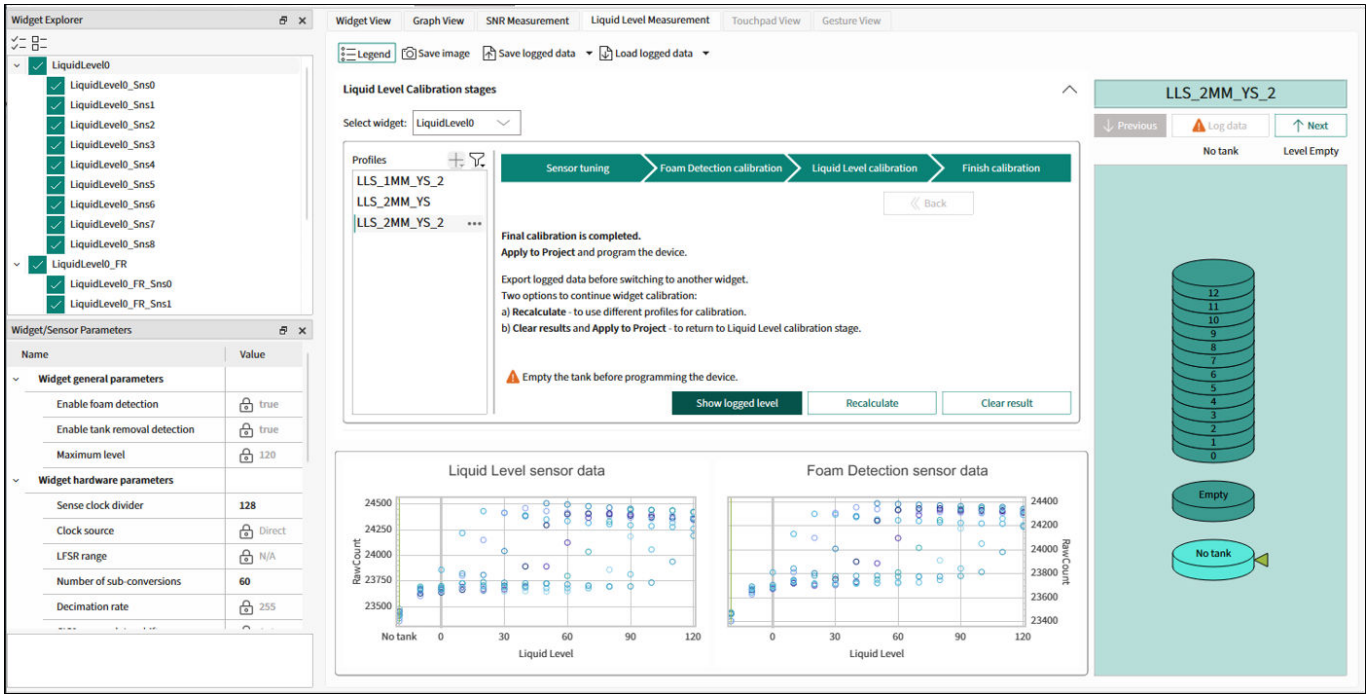
11 Liquid Level Measurement tab

The **Liquid Level Measurement** tab allows measuring high-resolution liquid levels based on the captured data. The tab provides the user interface to input the minimum step resolution and the total depth of the tank to measure. The Tuner calculates these parameters. Start with selecting a Liquid Level widget on Widget Explorer pane.

For a detailed description of the liquid level measurement and calibration process, refer to AN239805.

The toolbar is described in Tabs toolbars.

The tab contains the following panes: Liquid Level Calibration, Graph chart (LL), and Illustration of Liquid Level calibration.



The **Liquid Level Measurement** tab allows you to:

- measure the liquid level for different profiles for
  - Liquid Level widget
  - Liquid Level with foam detection – check **Enable foam detection** in the CAPSENSE™ Configurator (Widget details tab→Widget general parameters)

  The procedure is described in the Liquid Level Calibration and Calibration stages.
- detect the presence or absence of liquid with the Liquid Presence widget
- detect whether the tank has been removed – check **Enable tank removal detection** in the CAPSENSE™ Configurator (Widget details tab→Widget general parameters).

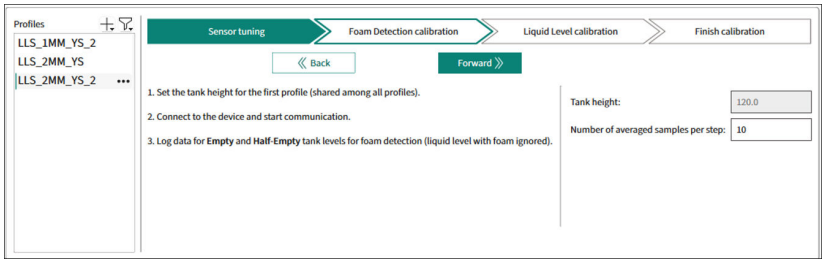
  ***Note**: The tank removal detection option enabled for the Liquid Level widget will also apply to the Liquid Level with foam detection.*

***Note**: Any changes in the configuration require re-programming the device before calibration.*

11.1 Liquid Level Calibration

The **Liquid Level Calibration pane** contains the **Profiles** tree and the stages to perform the Liquid Level calibration: **Sensor tuning, Foam Detection calibration, Liquid Level calibration** , and **Final calibration**. See Calibration stages for detailed instructions.

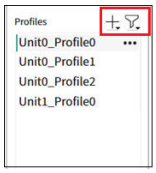
11 Liquid Level Measurement tab



Select widget: Select a widget from the list of available widgets.

Profiles

On the **Profiles** pane, you can add a new profile with the same or a new unit using the icons

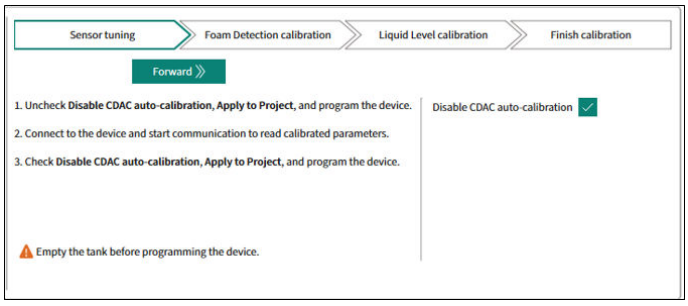


and sort the profiles by clicking "**...**".

Use the **Back** and **Forward** buttons to switch between the stages.

11.1.1 Calibration stages

Stage 1: Sensor tuning



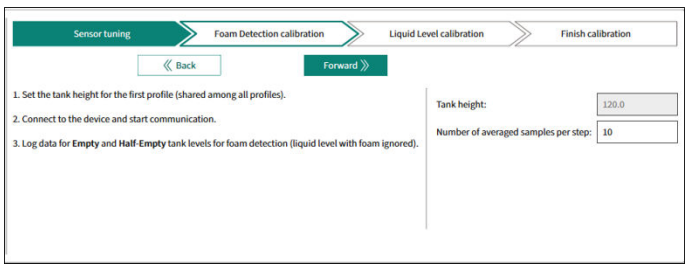
Instructions:

1. Uncheck **Disable CDAC auto-calibration**, **Apply to Project**, and program the device.
2. Connect to the device and start communication to read calibrated parameters.
3. Check Disable CDAC auto-calibration, **Apply to Project**, and program the device.

The parameter: **Disable CDAC auto-calibration**

Warning: "Empty the tank before programming the device."

Stage 2: Foam Detection calibration



11 Liquid Level Measurement tab

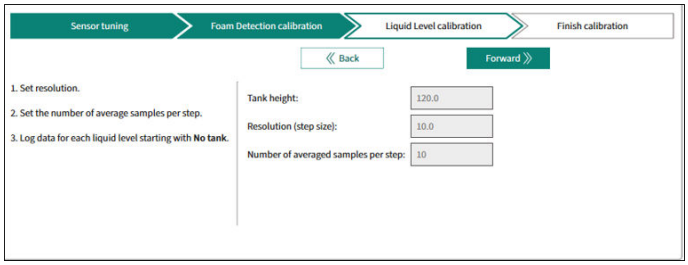
Instructions:

1. Set the tank height for the first profile (shared among all profiles).
2. Connect to the device and start communication.
3. Log data for each liquid level starting with **Empty** and **Half-Empty** tank levels for foam detection.

The parameters:

- **Tank height** – The height of the tank. Range: 30-1000.
- **Number of average samples per step** – The number of samples per measurement level. Range: 1-1000.

Stage 3: Liquid Level calibration



Instructions:

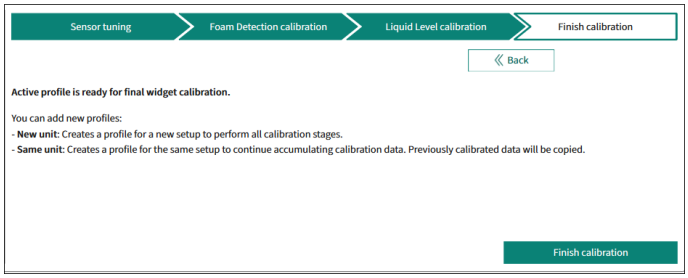
1. Set the resolution.
2. Set the number of average samples per step.
3. Log data from each level starting with **Empty** tank.

The parameters:

- **Tank height** – The height of the tank. Range: 30-1000.
- **Resolution (step size)** – The resolution of measurement. Range: 0.1-1000.
- **Number of average samples per step** – The number of samples per measurement level. Range: 1-1000.

Stage 4: Finish calibration

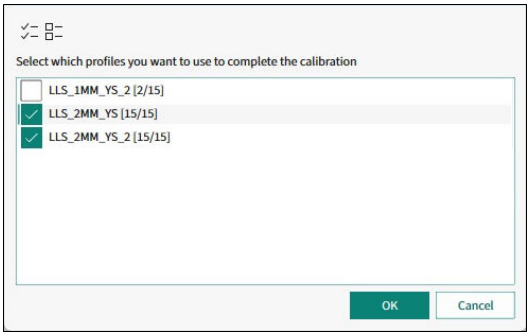
The active profile is ready for the final widget calibration.



Instructions:

- You can add new profiles:
  - New unit: Creates a profile for a new setup to perform all calibration stages.
  - Same unit: Created a profile for the same setup to continue accumulating calibration data. Previously calibrated data will be copied.

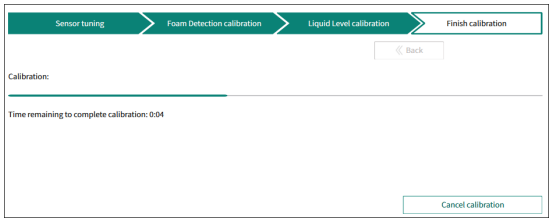
  The procedure is described in the Liquid Level Calibration and Calibration stages.
- Click the **Final calibration** button. The dialog "Calibration options" displays.



***Note**:* *A profile is considered valid if data is logged for:*

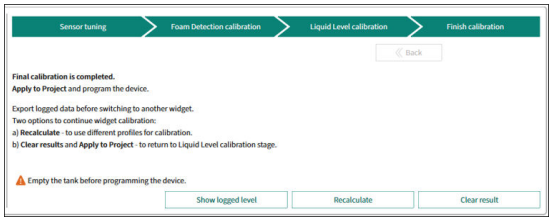
- ***Empty** and one level*
- ***No tank** if the tank removal detection is enabled*

- Select profiles.
- The Tuner starts the **Calibration** process.



Stage 5: Final calibration completed

The **Final calibration** stage is completed.



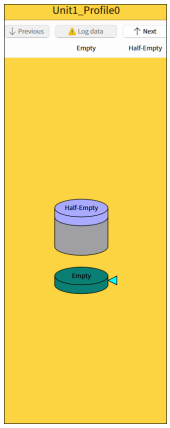
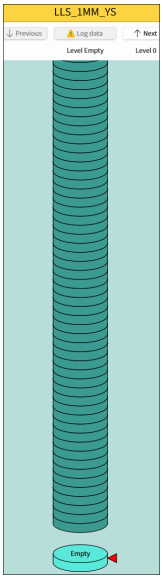
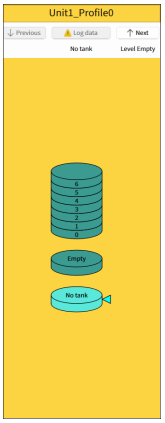
Instructions:

- **Apply to Project** and program the device.
- Export logged data before switching to another widget.
- The options to continue the widget calibration:
  - **Recalculate** – to use different profiles for calibration.
  - **Clear results** and **Apply to Project** – to return to the **Liquid Level calibration** stage.
- Warning: "Empty the tank before programming the device."
- **Show logged level**, **Recalculate**, **Clear results** – use these buttons for relevant actions.

11.1.2 Illustration of Liquid Level calibration

This pane is the graphical illustration of the calibration process. The current level (tank) is indicated by the triangle.

The images reflect the tank filling at different stages: **Empty**, **Half-Empty**. If the tank has been removed, **No Tank** image displays.

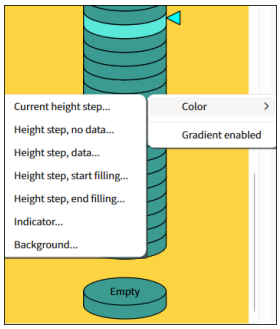
Foam Detection calibration	Liquid Level calibration	No tank detected
		

The pane contains buttons to execute the calibration process and a visual element.

- **Previous** – Allows switching to the previous level but does not log data automatically. Disabled when there is no previous level.
- **Log data** – Reads data from the device to the current level. After data is successfully read, the level automatically moves to the next one. The number of the level under the **Log data** button corresponds to the liquid level of the tank.
- **Next** – Allows switching to the next level.

Use the context menu to:

- set different colors for logged/not logged levels and background
- enable gradient



To see the height and data values of a level, hover over it.

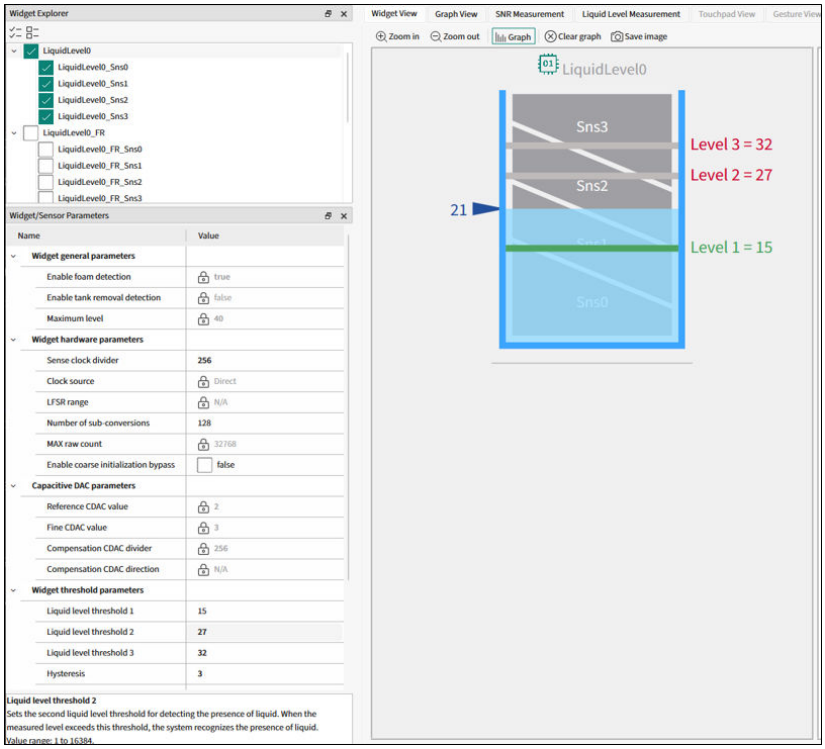
11.2 Multi-level liquid presence detection

The Tuner can represent several detection levels.

The **Multi-level detection** parameter in the CAPSENSE™ Configurator defines the number of liquid levels the sensor monitors. Select one of the Liquid level thresholds or disable the feature.

A single, global **Hysteresis** value is applied to all active thresholds to prevent signal oscillation.

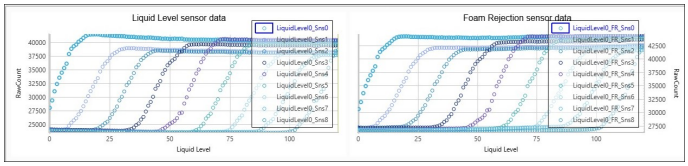
***Note**: Enter the thresholds in the ascending order (lowest to highest). A warning displays if the order is incorrect.*



After modifying **Liquid Level Threshold** or **Hysteresis**, click **Apply to Device**. The indicator will remain visible until the update is complete.

11.3 Graph chart (LL)

The **Graph** chart is empty until you start logging data and then the Liquid Level sensor data and Foam Detection sensor data charts get populated with Liquid Level and RawCount parameters data.



11.4 Save/Load logged data

Allows saving and loading the gathered raw counts during the measuring process to/from the *design.lls.csv* file residing along to the configuration file.

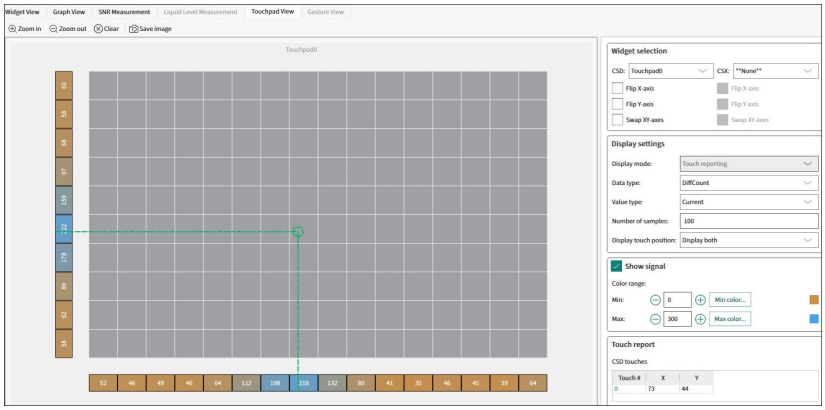
- **Export to another file...** – This option allows you to save gathered raw counts into a non-default user specified file
- **Import from another file...** – This option allows you to load raw counts data from a non-default user specified file.

12 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.

The toolbar is described in Tabs toolbars.

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



12.1 Widget selection

This pane consists of the configuration options for mapping physical touchpad orientation to the identical representation in the heatmap:

Use the CSD and CSX combo boxes to select a CSD or CSX touchpad to display it in the heatmap.

The options are as follows:

- **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.

12.2 Display settings

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

- **Display mode** – The drop-down menu with 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. In 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.
- **Display touch position** – Selects the touchpad from which data displays in the heatmap. The options:
  - Display only CSX
  - Display only CSD
  - Display both.

12.3 Show signal

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

- **Color range** – Defines the range of sensor signals within which the color gradient is applied. If the sensor signal is outside the range, the sensor color is either minimum or maximum out of the available color palette.

12.4 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 a touch is detected, the touch table is empty. When the two-finger detection is enabled for the CSD touchpad, 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 a touch is detected, the touch table is empty. The middleware supports simultaneous detection up to three touches for the CSX touchpad touch, so the touch table displays all the detected touches.

12.5 Hot keys

Keyboard shortcuts	Commands	Same as
Alt + N	Select a color for the minimum value.	Click the **Min color...** button.
Alt + X	Select a color for the maximum value.	Click the **Max color...** button.
Alt + B	Select a color for the text.	
Ctrl + K	Set the next scatter shape for the CSD mark.	
Ctrl + L	Set the previous scatter shape for the CSD mark.	
Alt + +	Increase the line width/mark radius.	
Alt + -	Decrease the line width/mark radius.	
Alt + 0 … Alt + 7	Select a color for the CSX position with ID=0…7.	Double-click the **ID** column cells of the CSX touches table.
Alt + 8	Select a color for the CSD touch position.	Double-click a cell in the **Touch #** column of the CSD touches table.
Ctrl + C	Clear the heatmap.	Click the **Clear** button on the local toolbar.
Ctrl + +	Zoom in the heatmap.	Click the **Zoom in** button on the local toolbar.

12 Touchpad View tab

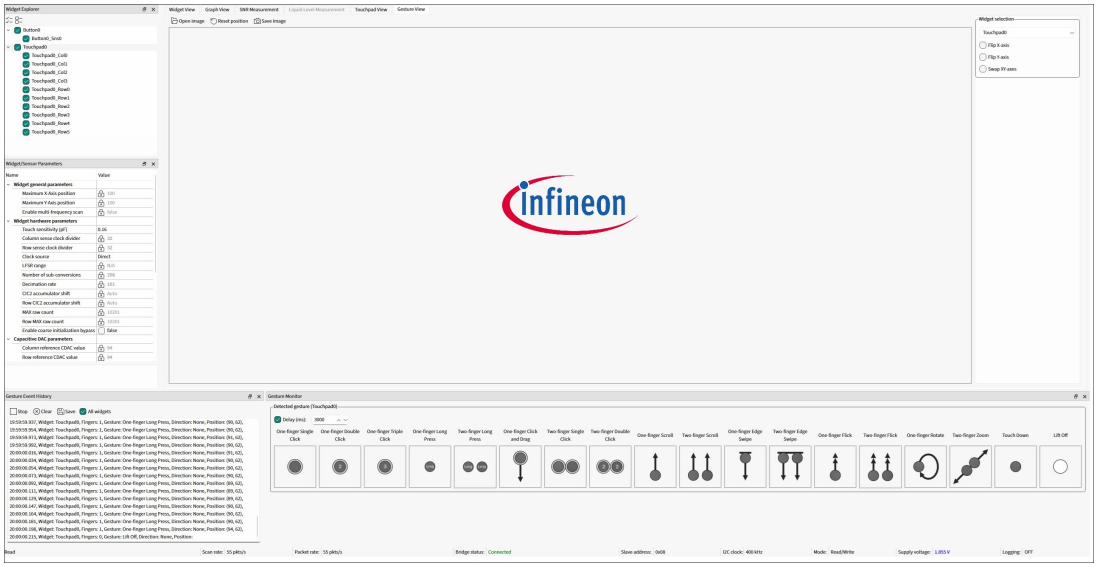
Keyboard shortcuts	Commands	Same as
Ctrl + -	Zoom out the heatmap.	Click the **Zoom out** button on the local toolbar.
Esc	Cancel zoom.	

13 Gesture View tab

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

The toolbar is described in Tabs toolbars.

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



***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.*

13.1 Widget selection

This pane consists of the configuration options for mapping physical touchpad orientation to the identical representation in the heatmap.

Use the combo box to select a widget to display the gesture you need.

The options are as follows:

- **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 the orientation of a physical widget.
- **Swap XY-axes** – Swaps the X- and Y-axes to match the orientation of a physical widget.

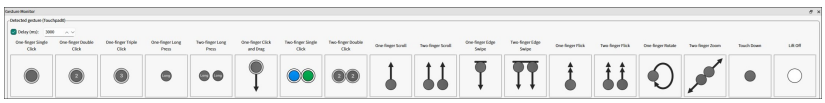
13.2 Image pane

An image reacts to the detected gestures (scroll, zoom, and rotate) and moves around the pane. You can change the image by clicking the **Open image** button or by using drag and drop.

You can reset the image view by clicking the **Reset position** button or pressing the **Esc** key.

13.3 Gesture Monitor

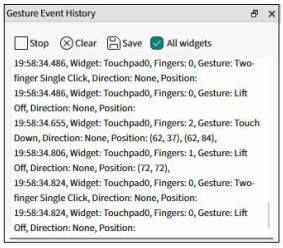
Provides visual indication for a detected gesture. It shows gestures for one widget at a time. The widget can be selected in the Gesture View tab.



If the **Delay** check box 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.

13.4 Gesture Event History

Logs the detected gestures information.



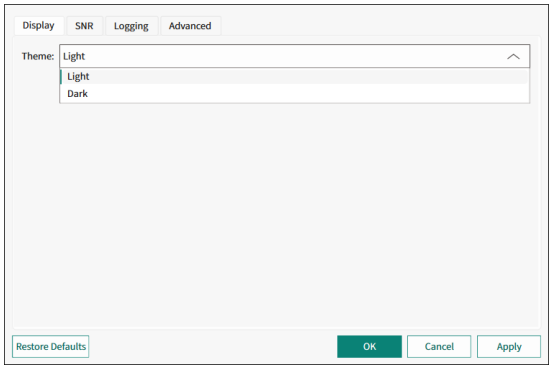
The toolbar options:

- **Start/Stop** – Starts/stops recording the history.
- **Clear** – Deletes the recorded data.
- **Save** – Saves the recorded data to the specified **.txt* or **.csv* file.
- **All widgets** – Check this to record the history for all widgets. Otherwise, only the history for the widget selected in the Gesture View tab will be recorded.

14 Tuner Configuration options

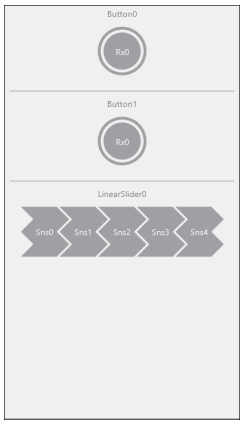
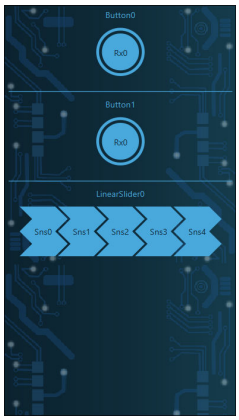
The Tuner application allows setting different configuration options with the options dialog. The settings are divided into groups.

14.1 Display options



14.1.1 Theme

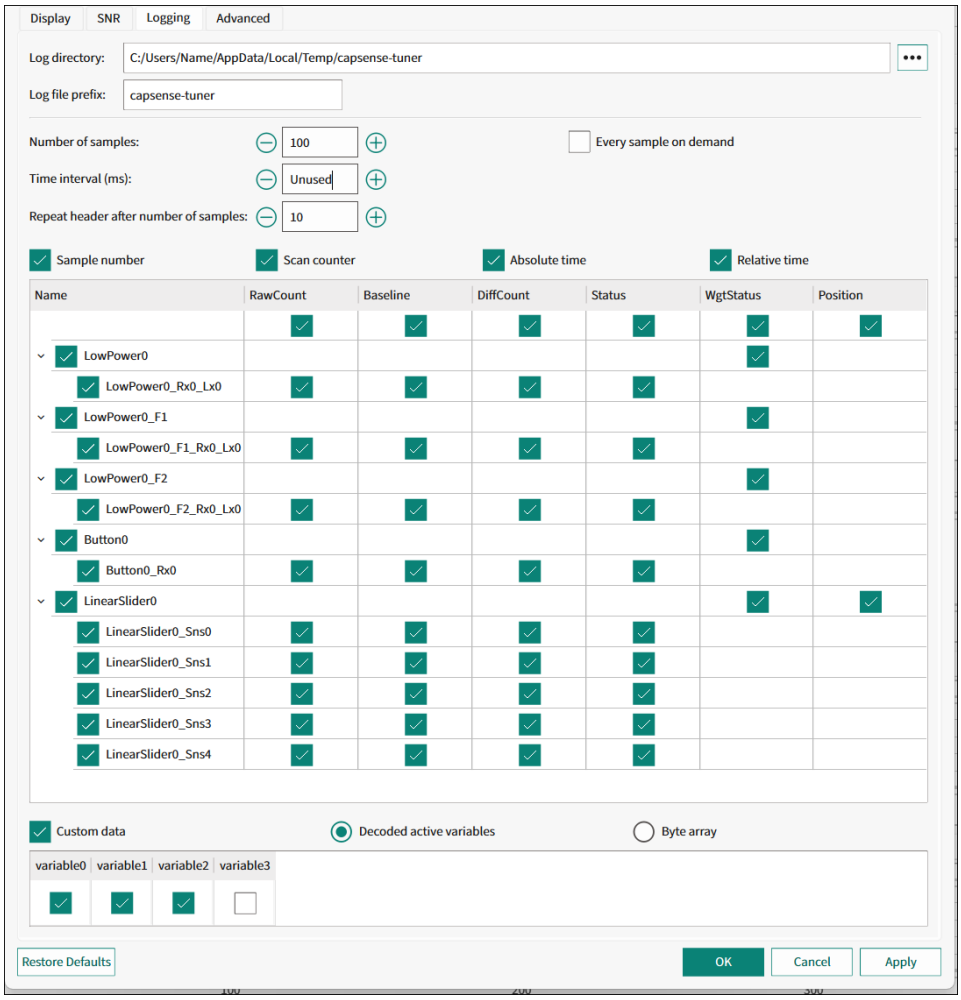
Defines the visual style of widgets.

Light	Dark
	

14.2 SNR

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

14.3 Logging options



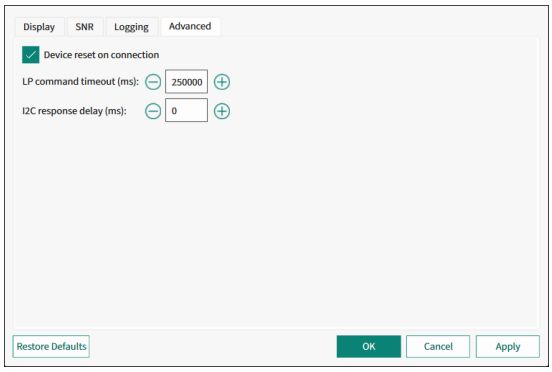
Fields:

- **Log directory** – Browse for the application directory where the CAPSENSE™ Tuner log files are located.
- **Log file prefix** – The prefix name is the user's input for the log file name. For example, *<prefix_name>-<time_stamp>.csv*.
- **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.

  Custom Data Settings

  You can enable or disable **Custom data** logging. Select between **Decoded active variables** and **Byte array**. In Decoded mode, the selected variables are logged as separate CSV columns. In Byte array mode, store each byte in a separate column and in the decimal format. Custom data variables cannot be changed during active logging, so the action for changing them is disabled while logging is running.

14.4 Advanced options



Fields:

- **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 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 Deep Sleep and the operation succeeds. Default: 250000 ms.
- **I2C response delay (ms)** – Configures a 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.

15 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 macOS 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.*

16 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.

16 Version changes

Version	Change descriptions
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.
	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.
6.20	Improved the auto-calibration feature for the MSCLP platform.
	Added support of CapDAC Auto-Dithering.
	Improvement – only relevant gesture parameters display.
	Added the ability to switch to UART async mode for LP devices.
	Added the editable prefix to the log file.
	Updated OpenOCD logging.
	Added displayed Sensor/Electrode capacitance.
6.30	Minor back-end changes.
6.40	Minor back-end changes.
7.10	Added support for ISX sensing method.

16 Version changes

Version	Change descriptions
	Added support for Liquid Level widget.
8.00	Added support of the PSOC™ 4100T Plus device family.
	Added scaling CDAC parameters for widgets that use the CSD sensing method.
8.10	Added the Power button control.
9.0	Added multiple sensing profiles support for Liquid Level widget.
10.0	Added detection of: • liquid presence/absence • tank removal
	Introduced additional gestures support.
10.10	Improved the foam rejection algorithm for liquid level sensing.
11.0	Extended the Tuner logging capabilities.
	Added the option to visualize thresholds in the Tuner.
	Added multi-level liquid presence detection.
12.0	Added the option to visualize and configure Custom Data.
	Improved the linearity of the liquid level sensing.

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.
*M	2024-02-13	Updated to version 6.20. Added the Hot keys section.
*N	2024-09-27	Updated to version 6.30.
*O	2024-12-05	Updated to version 6.40.
*P	2025-02-12	Updated to version 7.10.
*Q	2025-03-28	Updated to version 8.0.
*R	2025-09-04	Updated to version 8.10.
*S	2025-10-06	Updated to version 9.0.
*T	2025-12-17	Updated to version 10.0.
*U	2026-03-02	Updated to version 10.10.
*V	2026-05-22	Updated to version 11.0.
*W	2026-08-28	Updated to version 12.0.

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2026-08-28

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2026 Infineon Technologies AG

All Rights Reserved.

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-edk1712080553542

Important notice

Products which may also include samples and may be comprised of hardware or software or both ("Product(s)") are sold or provided and delivered by Infineon Technologies AG and its affiliates ("Infineon") subject to the terms and conditions of the frame supply contract or other written agreement(s) executed by a customer and Infineon or, in the absence of the foregoing, the applicable Sales Conditions of Infineon. General terms and conditions of a customer or deviations from applicable Sales Conditions of Infineon shall only be binding for Infineon if and to the extent Infineon has given its express written consent.

For the avoidance of doubt, Infineon disclaims all warranties of non-infringement of third-party rights and implied warranties such as warranties of fitness for a specific use/purpose or merchantability. Infineon shall not be responsible for any information with respect to samples, the application or customer's specific use of any Product or for any examples or typical values given in this document.

If this document is marked as 'Preliminary', 'Target', 'Draft' or 'Proposal', Infineon reserves the right to change all information given in this document at any time without notice. Subject to the development and release of a Product for series supply by Infineon, the technical specifications of the Product are set forth in the relevant datasheet provided by Infineon without such marking.

The data contained in this document is exclusively intended for technically qualified and skilled customer representatives. It is the responsibility of the customer to evaluate the suitability of the Product for the intended application and the customer's specific use and to verify all relevant technical data contained in this document in the intended application and the customer's specific use. The customer is responsible for properly designing, programming, and testing the functionality and safety of the intended application, as well as complying with any legal requirements related to its use.

Unless otherwise explicitly approved by Infineon, Products may not be used in any application where a failure of the Products or any consequences of the use thereof can reasonably be expected to result in personal injury.

However, the foregoing shall not prevent the customer from using any Product in such fields of use that Infineon has explicitly designed and sold it for, provided that the overall responsibility for the application lies with the customer. Infineon expressly reserves the right to use its content for commercial text and data mining (TDM) according to applicable laws, e.g. Section 44b of the German Copyright Act (UrhG). If the Product includes security features:

Because no computing device can be absolutely secure, and despite security measures implemented in the Product, Infineon does not guarantee that the Product will be free from intrusion, data theft or loss, or other breaches ("Security Breaches"), and Infineon shall have no liability arising out of any Security Breaches.

If this document includes or references software:

The software is owned by Infineon under the intellectual property laws and treaties of the United States, Germany, and other countries worldwide. All rights reserved. Therefore, you may use the software only as provided in the software license agreement accompanying the software.

If no software license agreement applies, Infineon hereby grants you a personal, non-exclusive, non-transferable license (without the right to sublicense) under its intellectual property rights in the software (a) for software provided in source code form, to modify and reproduce the software solely for use with Infineon hardware products, only internally within your organization, and (b) to distribute the software in binary code form externally to end users, solely for use on Infineon hardware products. Any other use, reproduction, modification, translation, or compilation of the software is prohibited.

For further information on the Product, technology, delivery terms and conditions, and prices, please contact your nearest Infineon office or visit www.infineon.com.