

PSoC® 4 - Capacitive Liquid-Level Sensing

Author: Greg Verge

Associated Part Family: CY8C42xx

Associated Code Examples: [CE202479](#)

Related Application Notes: [AN85951](#)

To get the latest version of this application note, or the associated project file, please visit <http://www.cypress.com/AN202478>.

AN202478 shows how to use PSoC 4 devices, CapSense® technology, and capacitive sensors to measure the depth or presence of water-based liquids in nonconductive containers. Sensors located on or near the container exterior provide real-time reporting of liquid height or percent remaining without liquid contact. Options exist to use low-cost sensor materials while still providing high-precision measurements.

Contents

1	Introduction.....	1	3.4	Sensor Pattern Choice.....	7
1.1	Beginner's Resources.....	2	3.5	Segmented Sensors	7
2	System Description.....	2	3.6	Differential Sensors	10
2.1	Block Diagram	2	4	Sensor Layout	13
2.2	Hardware Requirements.....	2	4.1	General Layout Considerations	13
3	System Theory	3	4.2	Segmented Sensor Pattern Considerations...	14
3.1	CapSense Basics	3	4.3	Differential Sensor Pattern Considerations....	14
3.2	Environmental Effects.....	4	5	Summary	14
3.3	CapSense Component Setup	5		Worldwide Sales and Design Support.....	16

1 Introduction

Liquid-Level Sensing (LLS) detects the presence and level of liquid in a container without any physical contact. There are various types of liquid-level sensors such as capacitive, mechanical float, inductive, magnetic, Hall effect, optical, acoustic density, and ultrasonic; each has advantages and disadvantages. Capacitive liquid-level sensing has become popular due to its low cost, high reliability, low power, sleek aesthetics, and seamless integration with existing control architectures.

Cypress's PSoC 4 devices support liquid-level sensing with resolution down to 1 mm. Capacitive liquid-level sensing is provided through the use of the CapSense_CSD Component available in the free PSoC Creator™ Integrated Development Environment (IDE). The CapSense_CSD Component configures the on-chip CapSense peripheral hardware and provides required firmware for operation on PSoC 4 devices. The following key liquid-level sensing benefits are provided using CapSense:

- Non-contact measurement avoids contamination and cleaning problems.
- Sensors located on the exterior of a nonconductive liquid container simplify industrial design and improve product user experience.
- Optimized resolution and accuracy to support varying price points with a single, low-cost, base system
- Sensors may be constructed out of low-cost materials such as plastic substrates and conductive ink.

This application note provides details on the design of hardware, firmware, schematics, and BOM. See Code Example [CE202479](#) for example projects demonstrating the concepts of this application note. The code example uses the combination of Arduino-compatible [CY8CKIT-042 PSoC 4 Pioneer Kit](#) and [CY8CKIT-022 Liquid Level Sensing Shield](#).

This application note is for advanced users of Cypress's CapSense technology. If you are new to CapSense technology, see the [Getting Started with CapSense](#) design guide. To learn about the implementation of CapSense technology on PSoC 4 devices, see the [PSoC 4 CapSense Design Guide](#).

This application note focuses on areas that differ from traditional finger-touch applications using CapSense. It is important to remember that no two end designs are the same and that the design rules provided do not have hard limits unless specifically stated. Taking one or more design parameters near or past the advised limits may still work with reduced but acceptable performance, while the same performance may be unsuitable for another application. Use the provided design recommendations as guidance, and perform validation under your design conditions.

1.1 Beginner's Resources

An overview of PSoC devices is available at www.cypress.com/psoc. The web page includes a list of PSoC device families, IDEs, and associated development kits. In addition, see the following documents to get started with PSoC 4 devices and CapSense technology:

- [AN79953 – Getting Started with PSoC® 4](#)
- [Getting Started with CapSense®](#)
- [PSoC® 4 CapSense® Design Guide](#)
- [CE202479 – Code Example for liquid-level sensing](#)
- [PSoC 101 Training series](#)

2 System Description

A capacitive liquid-level sensing system comprises two key design elements:

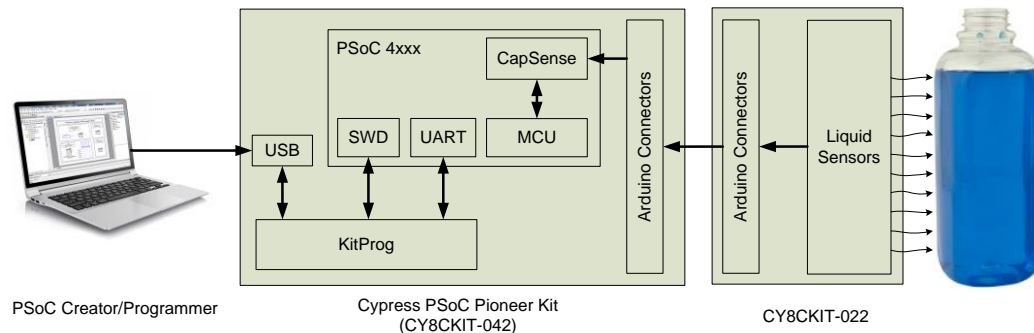
1. Capacitive sensor pattern to sense the liquid level
2. PSoC 4 device with the CapSense Component to measure the sensors and calculate the liquid level

This section provides a high-level overview of how they integrate into a larger system. Both design elements are described in detail in [Section 3](#).

2.1 Block Diagram

[Figure 1](#) shows the block diagram of the [CY8CKIT-022](#) Liquid-Level Sensing Shield kit, showing the simplicity of a PSoC 4-based liquid-level sensing design. Most designs require only a PSoC 4 device and a CapSense liquid sensor. Other system features can be integrated into the PSoC 4 device to further reduce system costs.

Figure 1. Liquid-Level Sensing Block Diagram



2.2 Hardware Requirements

To implement CapSense liquid-level sensing hardware, you need a PSoC 4 device and liquid-level sensors attached to a liquid container. Develop them as detailed in this application note or use those provided in the CY8CKIT-022 for an out-of-the-box, ready-to-run solution.

The [CY8CKIT-022](#) kit plugs into the [CY8CKIT-042 PSoC 4 Pioneer Kit](#) development platform from Cypress. Code Example [CE202479](#) provides liquid-level sensing example projects targeting the two kits.

[Figure 2](#) shows how the [CY8CKIT-022](#) kit connects to the [CY8CKIT-042](#) kit.

Figure 2. CY8CKIT-022 Connected to CY8CKIT-042



3 System Theory

This section provides a high-level description of the principle of capacitive sensing. For more CapSense operational and design details, see the [Getting Started with CapSense®](#) design guide and the [PSoC® 4 CapSense® Design Guide](#).

3.1 CapSense Basics

Capacitive liquid-level sensors are conductive pads or traces laid on a nonconductive material such as PCB, plastic, or glass. The intrinsic capacitance of the PCB trace, pads, and other sensor connections is called the sensor parasitic capacitance (C_P), as [Figure 3](#) shows.

When a target object such as water approaches the sensor, a small amount of liquid capacitance (C_L) is added to the C_P , as [Figure 4](#) shows. Liquid-level sensing involves measuring the increased capacitance when water is near the sensor.

Figure 3. Capacitance and Electric Field of a Capacitive Liquid-Level Sensor

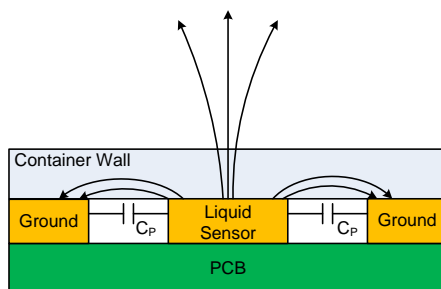
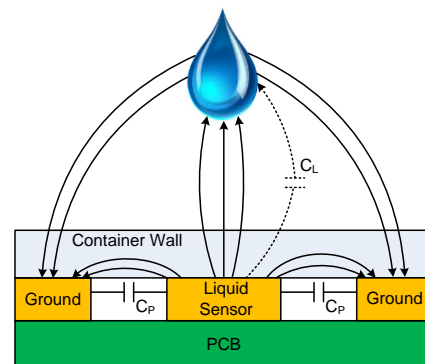


Figure 4. Added Capacitance (C_L) When Liquid Approaches a Capacitive Sensor



The PSoC 4 CapSense Component provided in the PSoC Creator IDE measures the capacitance by injecting a current into the sensor with a current Digital to Analog Converter (IDAC). A timer measures how long it takes the IDAC to charge the sensor's voltage to a reference voltage using a comparator. When the conversion is complete, the timer count value that measured the IDAC charge time is used as the raw sensor value used in calculations and is commonly referred to as the sensor count.

3.2 Environmental Effects

There are several important factors that can affect both the resolution and accuracy of liquid-level sensing. These effects are caused by variations in the sensor capacitance resulting in inaccurate measurements.

3.2.1 Temperature

Temperature fluctuations during operation have the most significant impact on performance. With finger-activated CapSense buttons (as compared to liquid level sensing), the sensor value while not pressed is tracked over time to account for any ambient temperature-induced offset. This compensated non-pressed value is called the baseline. In touch applications, this is possible because most of the time, the sensor is not touched. Additionally, the relatively short touch events provide a large instantaneous change in sensor values.

In liquid-level sensing applications, temperature compensation is more difficult. Unlike normally non-touched, finger-activated CapSense buttons, we cannot assume that the liquid sensor is uncovered with liquid as it may be covered to any level for any length of time. We must, therefore, compensate for temperature variation through the use of algorithms and optimized sensor designs. It is also important to accept that temperature has an impact on accuracy—design tradeoffs must be made to achieve an acceptable accuracy level. Specific temperature compensation methods are detailed in the sections describing each sensor pattern.

A secondary effect of temperature is condensation. A liquid that is significantly colder than the ambient air temperature may cause condensation to form on the sensor surface. Condensation may cause a higher capacitance, which in turn causes an increased error. Condensation during low-temperature testing can be reduced by insulating the surface of the sensor. Another technique is to provide a small insulating air gap between the liquid container and the sensor substrate. The air gap should be no larger than 3 mm for best performance.

3.2.2 Parasitic Capacitance

Parasitic capacitance (C_P) is the unwanted capacitance that exists between parts of an electronic circuit simply because they are located close to each other. The CapSense Component measures the total capacitance, which is the total of parasitic and liquid capacitance ($C_{TOTAL} = C_P + C_L$) within the limits of its dynamic range. In systems where C_P is larger, the parameter of interest, C_L , is the smaller of the total measured signal. This reduces the system sensitivity to liquid presence and therefore total system accuracy.

The main components of parasitic capacitance in CapSense designs are trace capacitance and sensor capacitance. C_P is a nonlinear function of sensor size, trace length, trace width, and trace-to-trace spacing. There is no simple relation between C_P and PCB layout features, but general principles exist. Increases in sensor size, increases in trace length and width, and decreases in the annular gap all cause an increase in C_P . One way to reduce C_P is to increase the gap between the sensor and ground. Unfortunately, widening the gap between sensor and ground decreases noise immunity. More detailed layout guidance is provided in section 4.

3.2.3 Mechanical Variations

Mechanical variations within the system can take many forms but all result in a change to the sensors' C_P . There are two types of mechanical variations.

1. Static variation is generally caused by manufacturing tolerances in the sensor assembly, PCB, and sensor alignment to the liquid container. Static variations, if understood and controlled, are compensated for during manufacturing with the baseline calibration operation described in the sensor sections 3.5 and 3.6.
2. Dynamic variation is caused by changes during operation. These changes often manifest themselves as changes to C_P by changing the sensor's capacitor dimensions. Similar to temperature changes, dynamic variation is difficult to compensate for. It is best to design the system to minimize dynamic mechanical variation effects on the sensors.

The most common mechanical variation encountered is a change in the distance between capacitor plates, where the first plate is the sensor and the second plate is the liquid surface. This can be caused by air bubbles in the adhesive used to attach the sensors to the container—the bubbles grow and shrink with air pressure. Another cause occurs when the sensors are not directly attached to the liquid container allowing a changing air gap between the container wall and the sensor substrate. To reduce these factors, ensure that there are no air bubbles present and the mechanical design is sufficient to maintain accurate sensor alignment.

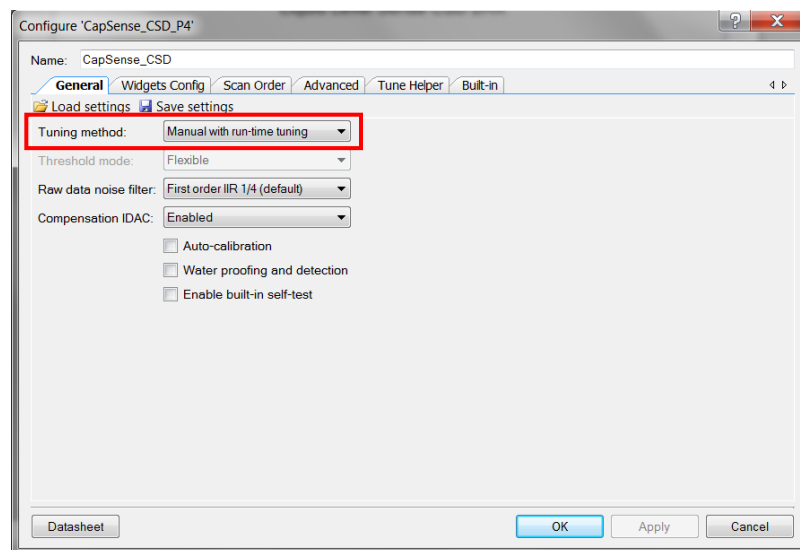
3.3 CapSense Component Setup

Liquid-level sensing uses the standard PSoC Creator IDE's CapSense_CSD Component to scan the liquid-level sensors. The Component functionality primarily supports touch applications and a touch-based baseline algorithm. Because liquid-level sensing requires a fixed baseline and custom liquid-level algorithms, we do not use the post-processing offered by the Component's standard touch Widgets or SmartSense™ auto-tuning capability. Instead, we use **Generic** sensors that provide only a raw capacitive value. Custom firmware is used to implement the liquid-level sensing algorithms. Parameters not discussed in this section either have no effect for liquid-level sensing or should remain at their default values. For more information on the Component parameters and their effects on tuning, see the [PSoC® 4 CapSense® Design Guide](#) and [PSoC 4 CapSense_CSD Component](#) datasheet.

Begin the CapSense Component setup by selecting **Manual with run-time tuning** as the tuning method on the **General** tab, as [Figure 5](#) shows. You can also select **Manual** tuning, but the tuning parameters are then hard-coded making tuning during development more time consuming. Do not select **Auto (SmartSense)** tuning, because it is not designed for liquid-level sensing and results in an incorrect tuning solution.

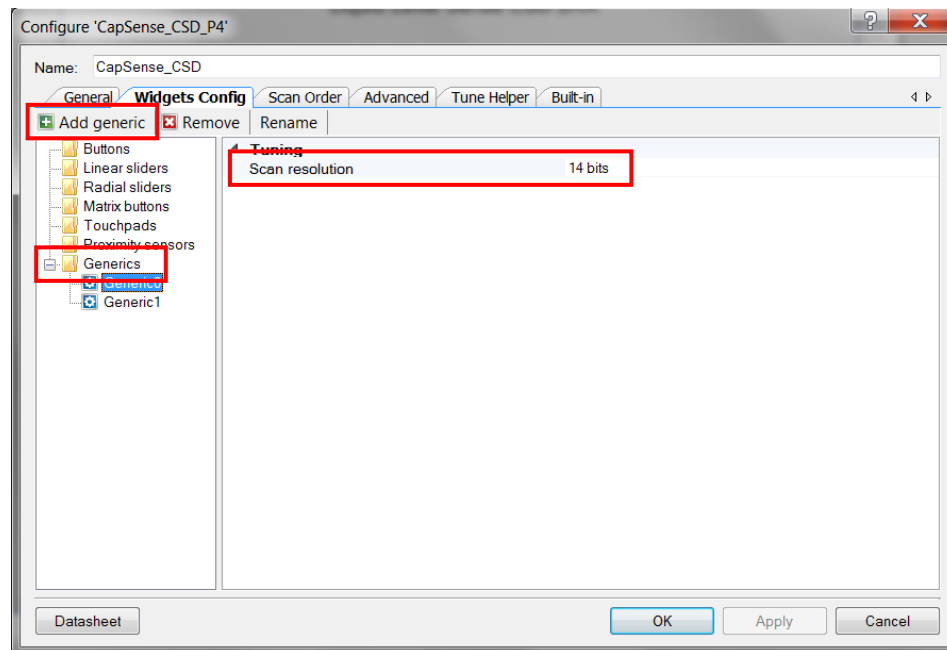
In most designs, the **Compensation IDAC** can be left **Disabled**. It is acceptable to **Enable** it to explore additional tuning options.

Figure 5. CapSense Component General Tab



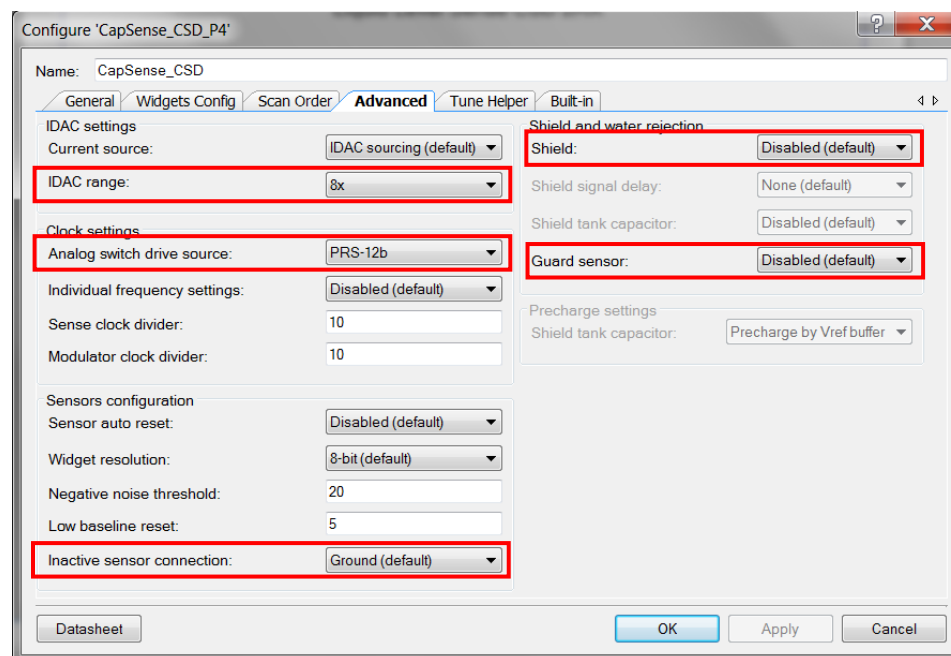
On the **Widgets Config** tab, select the **Generics** tree branch and click **Add generic** to add the number of sensor elements, as [Figure 6](#) shows. Generic sensors provide only a raw capacitance value without the touch-processing overhead. **Scan resolution** is the only tuning parameter that generic widgets have. Scan resolution determines the maximum resolution of the CapSense scan and therefore, the resulting scan time. In liquid-level sensing designs, a longer scan may be used to increase the liquid-level resolution. A good starting resolution is 14 bits; it can be modified later based on performance.

Figure 6. CapSense Component Widgets Config Tab



Only a few parameters on the Advanced tab are applicable to liquid-level sensing, as Figure 7 shows. The IDAC range may require modification from its default value of 4x if tuning is unable to converge on a valid raw count range. This is described in the tuning section of the [PSoC® 4 CapSense® Design Guide](#). The high C_P of the [CY8CKIT-022 Liquid Level Sensing Shield](#) kit requires the 8x IDAC range. The Analog switch drive source parameter should be set to PRS-12b to maximize the sensors noise immunity. The **Inactive sensor connection**, **Shield**, and **Guard Sensor** must all remain in their default settings as their purpose is to reduce the impact of water in touch designs. In liquid-level sense designs, we want to maximize the effect of water on the sensors.

Figure 7. CapSense Component Advanced Tab



3.4 Sensor Pattern Choice

Two sensor patterns are supported by this application note. See [Table 1](#) to determine which sensor pattern is best suited to a given set of design constraints.

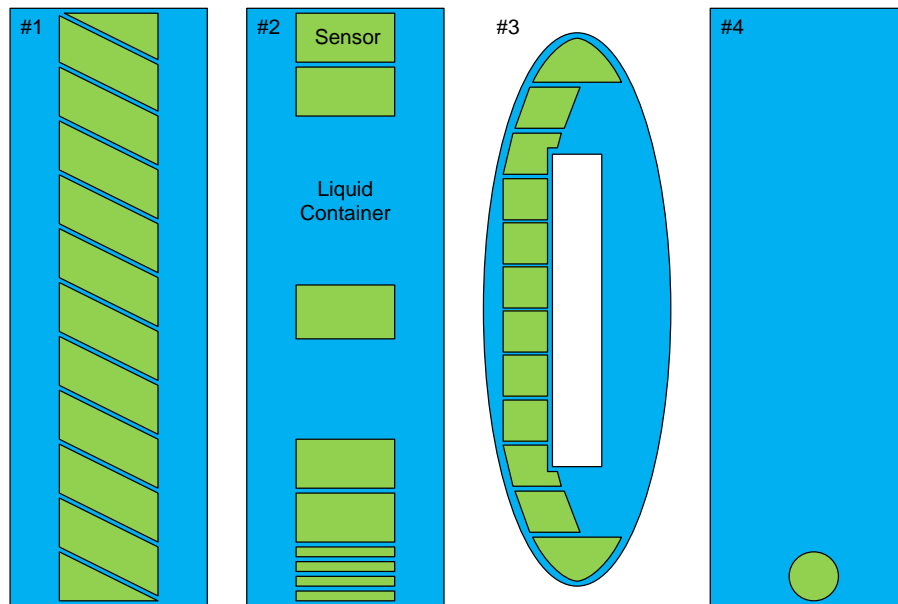
Table 1. Sensor Design Attributes

Sensor Design Attribute	Differential Sensor	Segmented Sensor
Number of required CapSense sensors	2	1 or more (Typically 10-20)
Sensor pattern constraints	Both sensors must be the same size and specific triangular shape	Each sensor may be a different size and shape
Resolution	Linear, ~1 mm	Determined by the total height of the container divided by the number of sensors. ≥ 1 mm
Accuracy	Typically, 20% accurate near empty improving to 5% accurate near full	Maximum accuracy is equal to the largest sensor's height and may be improved at critical levels by using shorter sensors (Typically, each sensor height is 5-10% of the total sensor)

3.5 Segmented Sensors

The most accurate and flexible sensor pattern is an array of sensor segments equally spaced from the top to the bottom of the liquid container. Each segment of the example sensor pattern #1 shown in [Figure 8](#) provides an incremental portion of the total liquid level. While the 'segmented' pattern requires more sensor elements and pins compared to the 2-sensor differential version, it allows liquid-level resolution to be customized for each design.

Figure 8. Segmented Sensor Examples



Sensor design and placement is very flexible allowing the sensor shape to fit within a given industrial design and provide resolution tailored to specific portions of the container.

- Sensor Height** – Each sensor's height is independent of the other sensors in the array. Changing the sensor height allows a small number of sensors to provide increased accuracy at critical levels such as empty and full, while providing decreased resolution in the middle of the container. The example sensor #1 shown in [Figure 8](#) uses a single height for most of the sensors and $\frac{1}{2}$ height sensors at the top and bottom to increase the accuracy at empty and full levels.

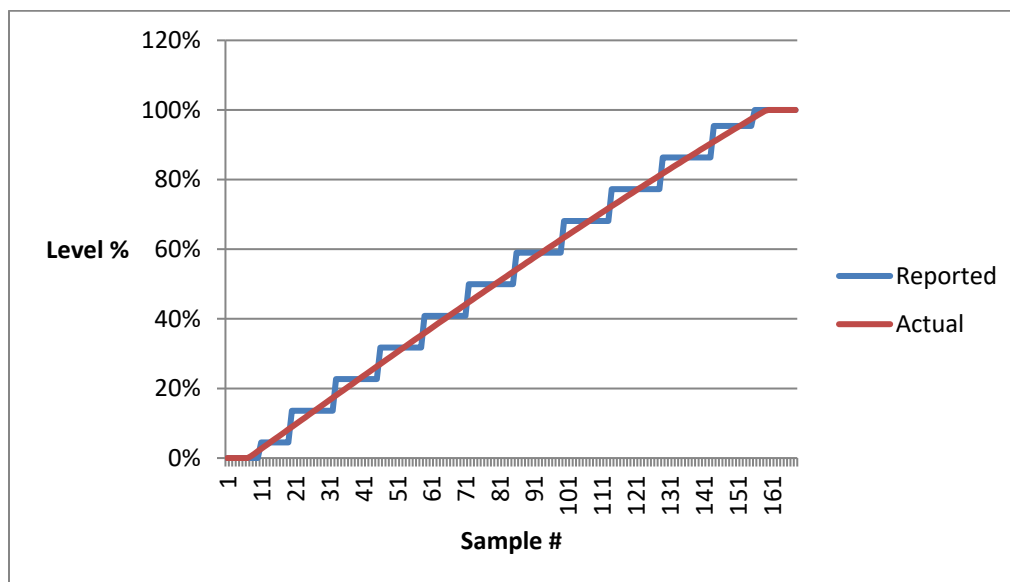
- **Sensor Shape** – Individual sensor shapes may be any geometric solid to assist in arranging the sensor array on complex shaped liquid containers or to account for industrial design constraints. Typical constraints include custom blow-moulded containers and pour spouts or mounting flanges that would otherwise block sensor placement as shown in example #3 of Figure 8.
- **Sensor Placement** – Sensors are usually evenly spaced along the container's height as shown in example #1 of Figure 8. If the application only needs liquid-level information at specific levels, sensors can be clustered at some levels and completely absent at other levels as shown in the example #2 of Figure 8. At the lowest limit of resolution, a single sensor can indicate a single critical liquid level, such as when the container is empty as shown in the example #4 of Figure 8.

3.5.1 General Operation

The segmented sensor pattern provides the highest accuracy across all operating conditions and is the best choice for most designs. The sensor pattern is segmented allowing each sensor element to accurately measure the liquid level in its limited range. Each sensor generates a binary output when it is half covered with liquid, by comparing the sensor counts with a reference value. The reported liquid level is the height of the highest sensor covered with liquid; if using an equally sized sensor array, it is the sum of the covered sensors multiplied by the height of each sensor.

In the example #1 sensor pattern shown in Figure 8, the top and bottom sensors are each half the height and area of the other sensors, thereby increasing the measurement resolution and accuracy at the empty and full limits. The two half-height sensors combined have the same height as one full-size sensor, thereby reducing the equivalent number of full-height sensors by one. Maximum liquid-level error is equal to 100% divided by the number of equally spaced sensors. The maximum error for the 12-sensor example is 100% divided by 11 (11 = 10 full height + ½ height + ½ height) which equals 9% for the full-height sensors. The typical error is half of the maximum error, which is 4.6% for the sensor shown in example #1. Due to the binary output of each sensor, the reported liquid level of the segmented sensors is stair-step compared to the actual liquid level as seen in Figure 9.

Figure 9. 12 Sensor Segmented Response



3.5.2 Liquid-Level Calculations

The calculation of liquid level from segmented sensors is very simple and comprises the following four main steps:

1. Measure static baseline of sensors.

The absolute value of each sensor is not important for determining whether liquid is present because the parasitic capacitance of the sensor and device-to-device variations can greatly impact the sensor's base value relative to other sensors and finished units. Prior to use, the raw value baseline of each sensor in an empty liquid container must be determined. The baseline values should be measured for each unit during manufacturing test and stored in nonvolatile memory. On subsequent bootups, the baseline values are recalled for use.

Liquid level is the most accurate when the operating conditions and temperature are most similar to the baseline measurement conditions. It is recommended to measure the baseline value at a temperature near the center of

the operating temperature range for improved accuracy. An optional method to increase the accuracy uses a temperature sensor to measure the temperature of the liquid or container and modify the baseline value within an experimentally derived range.

2. Remove static baseline from sensors counts.

During normal operation, each sensor's baseline value is subtracted from its raw value to find the difference counts. Difference counts are then limited to positive values and optionally scaled to a normalized value to account for sensors of different sizes and capacitance values.

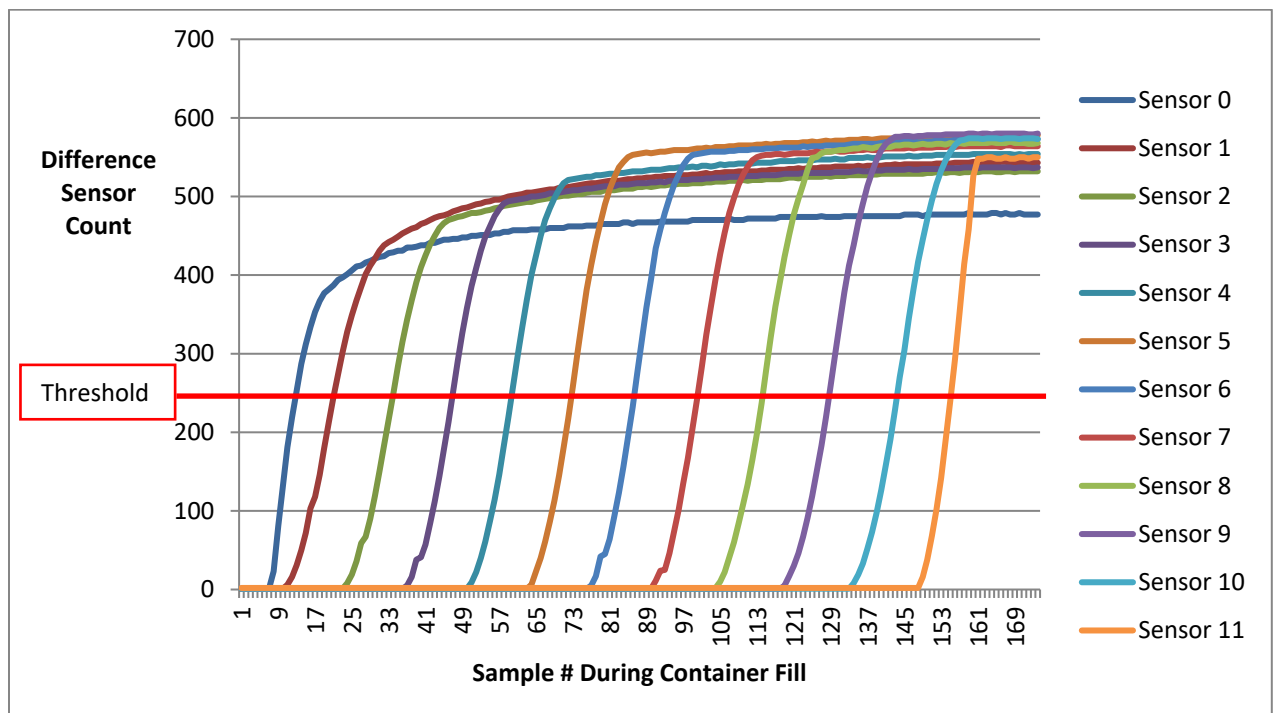
3. Determine which sensor segments are covered with liquid.

To determine whether a sensor is covered with liquid, its difference counts are compared to a threshold value as shown in Figure 10. Figure 10 shows the difference counts from each sensor as the container is filled. The data samples shown were recorded at one-second intervals while filling the container at a calibrated 1 mm per second rate. If the difference counts are above the threshold, it indicates that the sensor is covered and the binary state recorded. Hysteresis can optionally be added to the threshold to reduce the sensor state oscillation due to noise near the threshold value.

The threshold value should be set at $\frac{1}{2}$ the sensor difference counts when fully covered with liquid. The $\frac{1}{2}$ max threshold value ensures good level-sensing performance across a wide operating temperature range.

Sensor arrays with equally sized and spaced sensors can record the total number of sensors covered. Sensor arrays with irregularly sized or positioned sensors should record which sensor is the highest active sensor.

Figure 10. Segmented 12 Sensor Difference Count Versus Sample Number During Fill



4. Calculate the liquid height.

If the total number of equally sized sensors covered is known, then multiply the number of covered sensors by the standard sensor height. Standard sensor height equals the total sensor pattern height divided by the number of sensors. If the highest sensor covered is known, look up the height of that sensor from an array that contains the height of each sensor. Both methods result in a stair-stepped response as shown in Figure 9.

The calculated liquid height can optionally be filtered as required by the system response and converted to percent full by dividing the reported height by the total sensor pattern height.

3.5.3 Accuracy

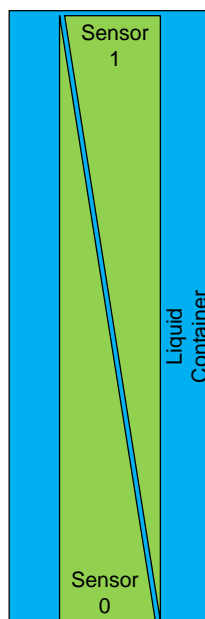
Because each sensor provides a binary contribution to the liquid level, the maximum error is limited to the highest partially covered sensor. The 12-sensor example design described in this section has a typical error of ± 7 mm ($\pm 4.6\%$) because the sensors are 14-mm high. The error has the following three components that combine for the total error:

1. Quantized error is formed by the resolution of the sensor's physical size and is \pm half of the sensors height. For example, a 10-mm tall sensor has a quantized error of ± 5 mm. Decreasing the sensor height improves the accuracy. The sensor height should always be greater than or equal to the container wall thickness to ensure sufficient signal level.
2. Temperature error is caused by temperature variations from the baseline temperature that change material permittivities. Temperature deviation in liquid-level sensing applications is typically less than 25% of the full-scale sensor value. Increasing the number of sensors decreases the temperature component of error.
3. Random errors are caused by parametric deviations and electronic interference in the system and difficult to calculate. Random errors can never be eliminated but can be minimized by following the design and layout recommendations found in the [PSoC® 4 CapSense® Design Guide](#).

3.6 Differential Sensors

The differential sensor method uses the ratio value of two sensors with triangular patterns and provides a low-cost method of determining the liquid level. The pattern shown in [Figure 11](#) looks similar to a backgammon game board and is therefore commonly referred to as the 'backgammon' pattern. The differential method provides a reduced accuracy compared to the segmented sensor but requires only two sensors for reduced sensor cost. The sensor pattern is composed of two sensors shaped so that the ratio of their values is equal to the percent value of the liquid level.

Figure 11. Differential Sensor Example

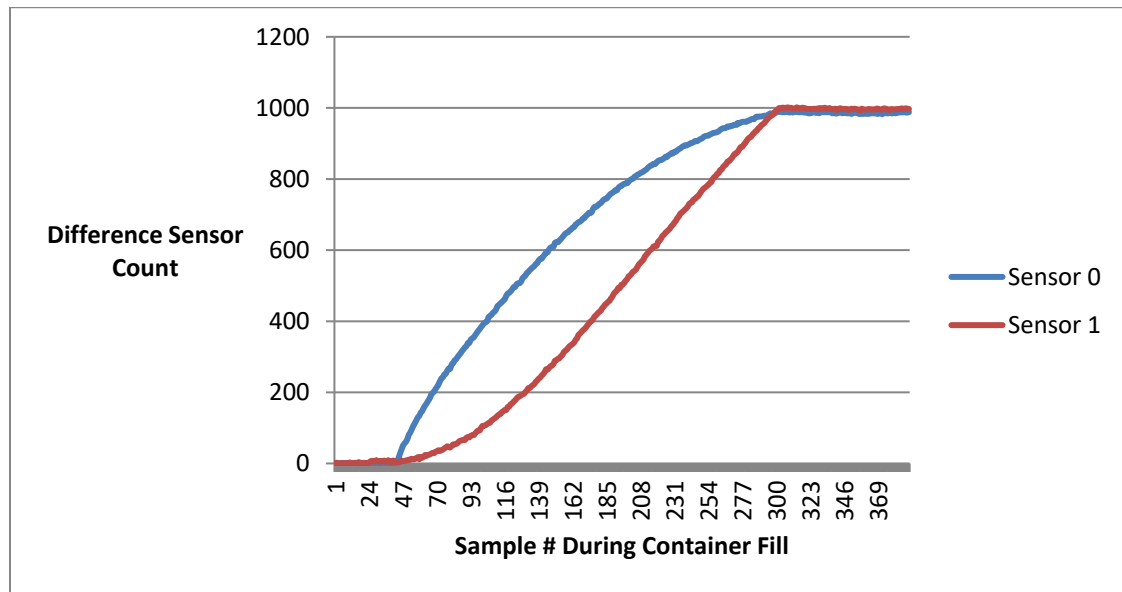


3.6.1 General Operation

The simplest sensor pattern is the differential pattern as only two sensors are required. The two key differences from the segmented pattern are reduced accuracy near the empty level and strict layout requirements for the two sensors.

The sensors are required to be right triangles with the bottom edge of Sensor0 and bottom point of Sensor1 at the empty liquid level while the opposite top edge and point must be at the full liquid level. While both sensors are covered to the same level with liquid, the total surface area covered by each sensor is very different. The sensor area covered by liquid directly relates to the total capacitance seen by each sensor. We can see in Figure 12 that Sensor0's value is increasing faster than Sensor1's value because it has more area in contact with the liquid just above the empty level than the small point of Sensor1. As we approach the full liquid level, we can see that Sensor0's rate of change decreases approaching the point while Sensor1's remains constant right up to the 100% level.

Figure 12. 2-Sensor Differential Values

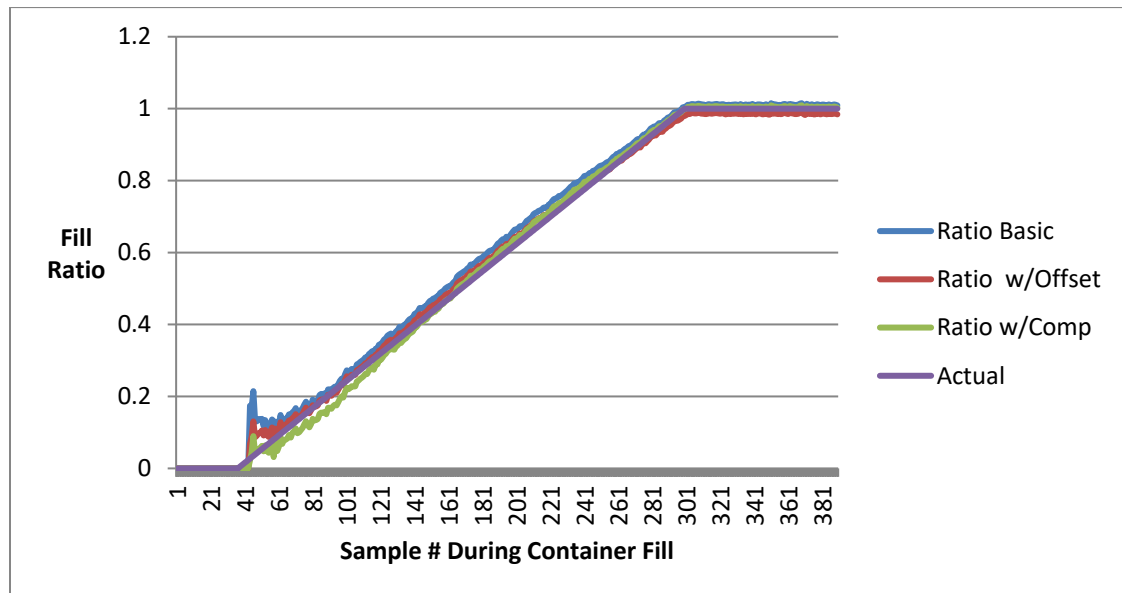


Because sensor count values are small near the empty level, any sensor error is magnified into a large error in liquid level. As the liquid level increases, the same error value becomes a smaller portion of the total value; therefore, the accuracy is best at the full level. So, the differential sensor should not be used in designs that require accurate determination of an empty container.

The differential nature of the sensors also requires that both sensors be of exactly the same size, have the same exposure to the liquid, and the same parasitic capacitance. Even an optimally constructed sensor pattern has errors impacting the calculated liquid level; therefore, a method of compensating the reported level is provided.

We can see in [Figure 13](#) that the calculated liquid level output is more linear than the segmented sensor pattern and there is a large error 'spike' near empty on the 'Ratio Basic' graph.

Figure 13. 2 Sensor Differential Response



3.6.2 Liquid-Level Calculations

The calculation of liquid level from differential sensors is a simple ratio calculation. There are four steps:

1. Measure the static baseline of sensors.

The baseline storage process and reason are the same as for the segmented sensor pattern.

2. Remove the static baseline from sensors counts.

During normal operation, each sensor's baseline value is subtracted from its raw value to find the difference counts. The difference counts are then limited to positive values.

3. Calculate the sensor ratio.

The sensor ratio is calculated as Sensor1 divided by Sensor0 when using the sensor pattern shown in [Figure 11](#). The resulting ratio is shown as the 'Ratio Basic' graph in [Figure 13](#).

Due to the small values on Sensor0, which is the denominator of our ratio equation, any error or offset on Sensor1 is magnified as can be seen with the 0.2 spike (20% error) in the ratio near the empty level. To reduce this error, we can add a small positive offset to Sensor0's value to minimize the effect of the error. By adding an offset of +25 counts to Sensor0's difference counts in the example data, the error is cut in half as 'Ratio w/Offset' shows in [Figure 13](#). The greater the offset value, the smaller the empty error becomes, but the greater the full error. The offset of +25 counts decreases the maximum ratio possible to 0.98 of the full liquid level, and is never reported as 100% full.

One additional special case checked for are divide by zero errors if Sensor0 = 0. To avoid this if Sensor0 = 0 then we skip the ratio calculation and set the liquid level to 0%.

4. Compensate the sensor ratio.

We saw in step 3 how the ratio error can be reduced, but there still may be original or offset-induced linearity issues with the reported liquid level. The 'Ratio w/Comp' graph in [Figure 13](#) shows the improvement in accuracy that can be attained by performing a slope and offset modification to the 'Ratio w/Offset' value. The combination of a small slope adjustment and offset further reduces the error near empty as well as allows the full value to reach a ratio of 1.0 (100%). The compensated ratio is calculated by multiplying the offset ratio with an experimentally derived constant and then adding an offset constant. In most designs, these constants will both be near 1.

The calculated liquid height can optionally be filtered as required by the system response and converted to height in millimeters by multiplying the reported ratio by the total sensor-pattern height.

3.6.3 Accuracy

Because of the small numbers used in the ratio calculation near the empty level, the calculated liquid level is least accurate near empty. The two-sensor example design described in this section has a typical error of $\pm 15\%$ (± 23 mm) near empty. The error has the following three components that combine for the total error:

1. Small number errors are caused by the large error produced when small numbers are used to calculate a ratio. For example, an error of 1 divided by 100 is a 1% error, while an error of 1 divided by 2 is a 50% error. This error is intrinsic to the differential design but can be reduced by adding a positive offset to the denominator sensor value. As the added offset increases, the error at the full level increases, which can be partially compensated for. The optimal offset and compensation slope equation constants must be experimentally determined for each design.
2. Temperature error is created by the sensor value offset caused by temperature deviation from the baseline temperature. Temperature deviation in liquid-level sensing applications is typically less than 25% of the full-scale sensor value. Temperature affects both sensors simultaneously; therefore, the temperature effect is less than 25% as the offsets partially cancel out.
3. Random errors are caused by parametric deviations and electronic interference in the system and difficult to calculate. Random errors can never be eliminated but can be minimized by following the design and layout recommendations provided in the [PSoC® 4 CapSense® Design Guide](#).

The ratiometric calculations require fraction values to maintain accuracy. While floating-point numbers could be used, they result in larger and slower code. To provide the required accuracy and avoid floating-point numbers and calculations, fixed precision values are used instead. If you are not familiar with fixed precision numbers and calculations, see articles on the subject at https://en.wikipedia.org/wiki/Fixed-point_arithmetic and https://en.wikipedia.org/wiki/Binary_scaling.

4 Sensor Layout

4.1 General Layout Considerations

Liquid-level sensor layout guidelines are nearly identical to the CapSense touch layout guidelines in the [PSoC® 4 CapSense® Design Guide](#), therefore only key highlights and differences are presented in this application note.

- The industrial design should ensure that the sensors and traces are isolated from conductive materials or sources of interference. Air gaps provide the best isolation from conductive objects as the relative permittivity of air is the lowest available at 1.0. An air gap of at least 5 mm provides excellent isolation when used with a hatched ground plane. As conductive objects approach closer to the sensors, the effectiveness of a hatched ground decreases as more of the sensors' field escapes between the hatched lines to the conductive object.
- The liquid container can be any nonconductive material. Materials with higher relative permittivity perform better by increasing the signal level of sensors. Material guidelines are the same as for touch application overlay materials shown in [Table 2](#).

Table 2. Relative Permittivity of Container Materials

Material	ϵ_r
Air	1.0
Glass (Standard)	7.6 – 8.0
Glass (Ceramic)	6.0
Polycarbonate (Lexan®)	2.9 – 3.0
Acrylic (Plexiglass®)	2.8
ABS	2.4 – 4.1

- Container wall thickness should be minimized, because thinner walls increase the sensor signal level. Container walls thinner than 5 mm give the best performance. Thicker walls can be used with increased scan times and possible decreases in accuracy.
- The sensor-to-container adhesive must be nonconductive and should have a high permittivity. 3M™ 467MP and 468MP transfer tapes provide excellent performance.
- Parasitic capacitance, C_P , should be minimized to maximize the sensors' response to liquids. The area of each sensor in the sensor pattern should be less than 3,000 mm². The space between sensors should be equal to the

container wall thickness or 1 mm, whichever is wider. Minimum sensors area is 25 mm² while optimal performance is typically found near 150 mm².

- Sensor and PCB board thickness should be greater than 0.25 mm to maximize the gap between the sensors and the ground plane.
- Sensor trace width should be no wider than 7 mil (0.18 mm) with a 10 mil to 20 mil (0.25 mm to 0.51 mm) clearance. Trace length should be minimized and ideally less than 30 cm.
- Sensor traces should be routed on the PCB side opposite the liquid surface to reduce interaction. Traces should also be routed at least 0.25 mm from switching signals and should ideally have a ground between them. If sensor traces must cross switching signals, they should cross at right angles.
- A ground plane must be used on the top and bottom of the PCB, wherever sensor traces are present, to increase sensor immunity to electronic interference. To limit sensor C_P , the ground planes over and under the sensors and traces must be hatched with a 17% fill (7-mil line, 70-mil spacing). All other areas should use a solid ground plane optimized for precision analog circuits.
- Power supply decoupling capacitors should be provided as recommended in the device datasheet to minimize noise in the high-sensitivity analog system of PSoC CapSense.
- GPIO pins used for sensors should have 560-Ω resistors in series with the sensors, placed as close to the pin as practical for increased EMI immunity.

4.2 Segmented Sensor Pattern Considerations

There are no additional requirements for segmented sensor patterns, as the sensors can be of any practical shape.

4.3 Differential Sensor Pattern Considerations

The differential sensor pattern has the following additional unique constraints to ensure performance:

- Both sensors must be as close in size as possible to provide matched C_P .
- Both sensor traces should have similar routing to provide matched C_P .
- Both sensors must be precisely aligned and mirrored right triangles, as [Figure 11](#) shows.
- The optimal height-to-width aspect ratio of each sensor is 8:1. Sensor aspect ratios should fall within the range of 16:1 to 1:1 with lower ratios providing greater accuracy.

5 Summary

This application note provided guidance on the design and implementation of capacitive liquid-level sensing. Two sensor patterns and algorithms were presented to allow optimization of the design to meet requirements.

Document History

Document Title: AN202478 - PSoC® 4 - Capacitive Liquid-Level Sensing

Document Number: 002-02478

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5037771	GJV	12/08/2015	New Application Note.
*A	5741512	AESATMP9	05/18/2017	Updated logo and copyright.
*B	6410619	GJV	12/13/2018	Minor updates to grammar and clarifications throughout document.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at [Cypress Locations](#).

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

[PSoC 1](#) | [PSoC 3](#) | [PSoC 4](#) | [PSoC 5LP](#) | [PSoC 6 MCU](#)

Cypress Developer Community

[Community](#) | [Code Examples](#) | [Projects](#) | [Videos](#) | [Blogs](#)
| [Training](#) | [Components](#)

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.



Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2015-2018. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spanion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

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

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