Please note that Cypress is an Infineon Technologies Company.

The document following this cover page is marked as "Cypress" document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content

The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers

Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

Objective

This example demonstrates liquid level sensing using capacitive sensors.

Overview

This code example demonstrates how to use PSoC™ 4, CapSense™ technology, and capacitive sensors to measure the depth or presence of water-based liquids in non-conductive containers. Sensors located on or near the container's exterior provide real-time reporting of liquid level. Many options exist to use low-cost materials to construct and integrate the sensors while still providing high-precision measurements.

The firmware continuously measures the data from sensors and outputs the calculated liquid level over a serial UART connection. PSoC 4 Pioneer development kits provide UART interface to the user's computer through a USB connection.

The two projects *LLS_CSD_2RX-042* and *LLS_CSD_12RX-042* demonstrate liquid-level sensing on 2-sensor (2RX) and 12-sensor (12RX) patterns using the CY8CKIT-022 CapSense Liquid Level Sensing Shield and CY8CKIT-042 PSoC 4 Pioneer Kit. Both projects are contained in this code example's workspace. These projects support water-based liquids using CapSense self capacitance. An update scheduled for 2016 will support additional liquids, higher-performance CapSense mutual capacitance scanning, and increased noise immunity.

Requirements

Tool: PSoC Creator 3.3 CP1 or later

Programming Language: C (GCC 4.9)

Associated Parts: PSoC 4 devices with CapSense

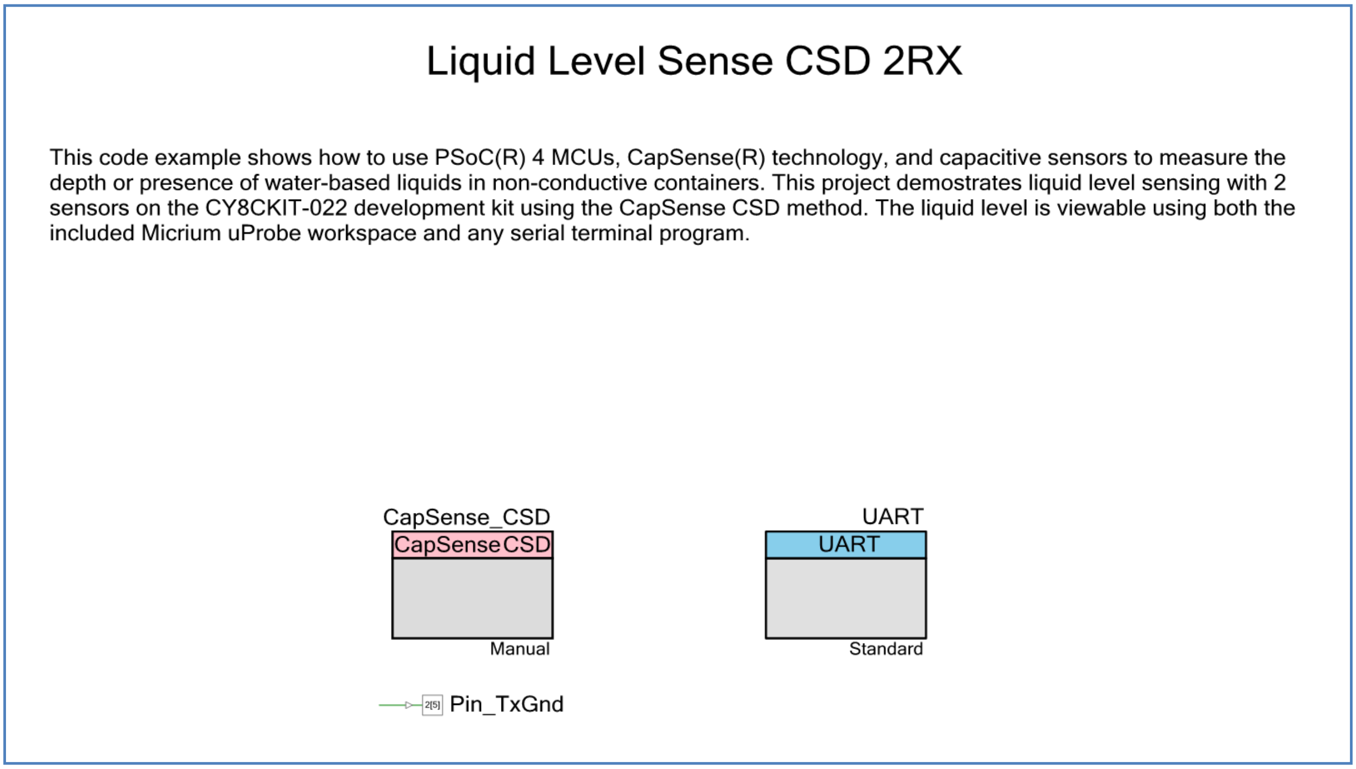
Related Hardware: CY8CKIT-022, CY8CKIT-042

Design

Figure 1**Error! Reference source not found.** shows the PSoC Creator schematic for the *LLS_CSD_2RX-042* code example. For additional information on liquid-level sensing theory, algorithms, sensor layout, tuning, calibration, and system design guidance, see Application Note AN202478 – PSoC® 4 - Capacitive Liquid Level Sensing. The projects feature the following Components:

- CapSense CSD Component for measuring the signal level on each of the liquid sensors. Liquid level is calculated after each scan is completed.
  - Self-capacitance scan method
  - 2 or 12 generic sensors, depending on the sensor used for custom liquid-level processing of raw values.
- UART Component for data viewing and command entry.
  - Standard 115200 baud rate, no parity, 8 data bits, 1 stop bits (N81) terminal interface.
  - Command input and data output

Figure 1. PSoC 4 2RX Sensor Liquid Level Sense Schematic



Design Considerations

The CapSense Component in the projects is tuned to work with the CY8CKIT-022 plus CY8CKIT-042 kit combination and the liquid container provided. The CY8CKIT-022 sensors can be attached to a user-supplied container, or other compatible hardware may be used, although re-tuning of the CapSense Component may be required. The CapSense_CSD Component can also add or remove sensors from the 12RX project to allow operation with custom sensors with a different number of CapSense sensor elements.

Project tuning and algorithms are optimized for the permittivity, surface tension, and conductivity of water. The default tuning will work with most human-consumable, water-based liquids. Liquids with properties that differ from water may require tuning, algorithmic, or sensor geometry changes outside the scope of this code example.

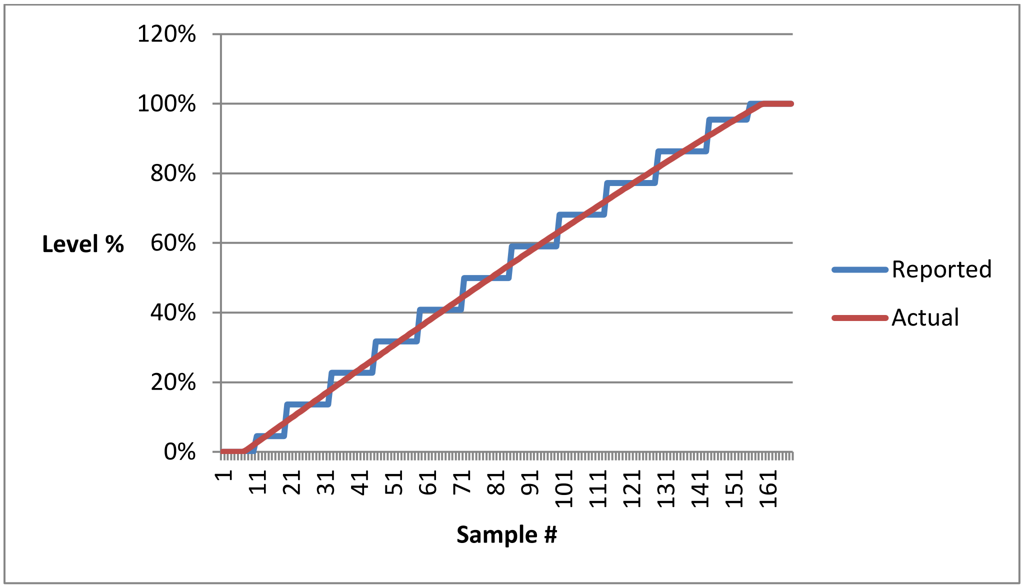
Sensor Selection

- 12RX Sensor

  The 12-sensor flexible PCB provides the highest accuracy across all operating conditions and is the best choice for most designs. The 12RX sensor is segmented allowing each sensor element to accurately measure the liquid level in its limited range.

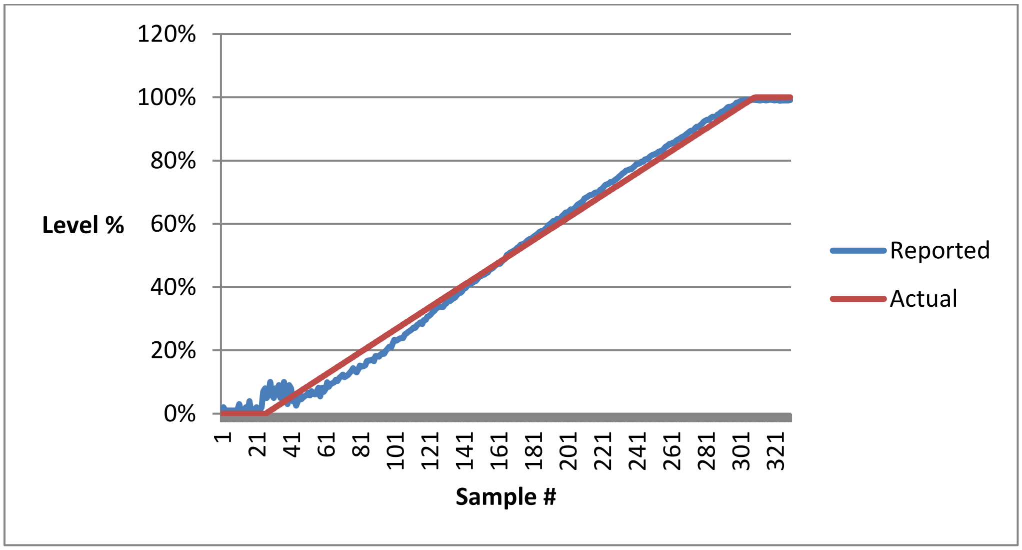
  Each sensor generates a binary output when it is approximately half covered with liquid. The reported liquid level is the sum of covered sensors multiplied by the height of each sensor. The top and bottom sensors are each of half the height of other sensors, thereby increasing the accuracy at the empty and full limits and reducing the sensor height by one. Maximum liquid level error is equal to 100% divided by the number or sensors – 1. The maximum error for the 12RX sensor is 9%. Typical error is 4.5%. The reported liquid level of the 12RX sensor is stair-step compared to the actual liquid level as seen in Figure 2

Figure 2. 12RX Sensor Response



- 2RX Sensor
- The 2-sensor flexible PCB provides reduced accuracy compared to the 12RX sensor but requires only two sensors for reduced sensor cost. The 2RX sensor is composed of two sensors shaped so that the ratio of their values is equal to the percent value of the liquid level.
- Any offset on the sensor values generates a liquid-level error that is greatest at low liquid levels. The typical error for the 2RX sensor is 15% and decreases as the container fills. Maximum error is determined by the system's response to temperature changes that impact the sensors response. The reported liquid level of the 2RX sensor is linear as seen in Figure 3.

Figure 3. 2RX Sensor Response



Hardware Setup

This code example is designed to run on a CY8CKIT-022 shield board mounted on a CY8CKIT-042 Pioneer kit. For the detailed kit board setup, see the corresponding CY8CKIT-022 kit user guide. Hardware setup consists of the following basic steps:

1. Mount the CY8CKIT-022 shield board on a CY8CKIT-042 Pioneer board.

2. Attach 2RX or 12RX flexible sensor to the container supplied, using the pre-attached adhesive. The sensor chosen must match the firmware programmed into the PSoC device.

3. Connect the flexible sensor to the shield board. Ensure there are no air pockets between the sensor and container as they can cause erratic sensor results if the distance between the sensor and liquid changes.

4. Connect the CY8CKIT-042 Pioneer board to a computer using a USB cable

Software Setup

This code example firmware supports serial terminal program computer interfaces. The UART interface outputs the current liquid level height and allows empty level calibration through a terminal emulator.

Serial Terminal

The documentation for setting up a terminal emulator for this example uses Tera Term but any terminal emulator software may be used that is configurable to the standard UART settings shown in Figure 5. Tera Term is open source and downloadable directly from the author's website at https://en.osdn.jp/projects/ttssh2/. All serial commands supported by the firmware are lower case and case sensitive.

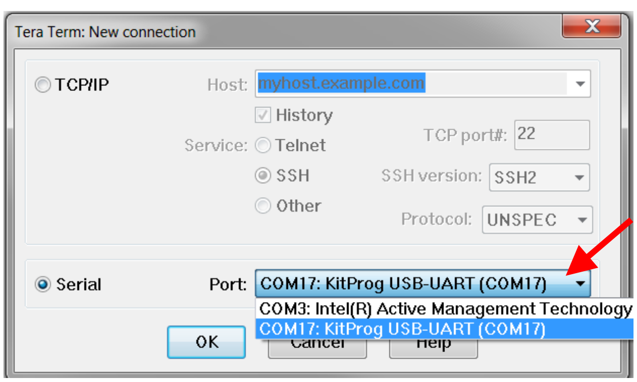
Supported serial commands are;

- `stop` – Stops outputting data over the serial connection
- `cal` – Stores empty container sensor values to EEPROM for calibration of future readings
- `basic` – Continuously outputs liquid level in millimeters (mm) and percent (%).
- `csv` – Continuously outputs intermediate computation values as well as liquid level in CSV format. The CSV format supports easy terminal emulator logging and data analysis with a spreadsheet or other tools.
- [**Enter**] – Outputs the next set of level values from the sample array
- `Reset` – Resets the sample array pointer to zero

1. Create new connection.

   Launch Tera Term and select **File** > **New connection**. Select **Serial** as the connection type and choose the **KitProg USB-UART** communication port (COM) as shown in Figure 4. The actual COM port number will vary between computers and USB ports. If multiple communication ports are listed, it can be helpful to disconnect and reconnect the development kit USB cable and look for the COM port that is disappears and reappears.

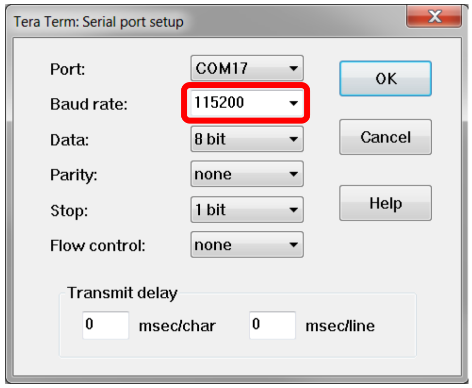
Figure 4. New Connection Creation



2. Setup serial port parameters.

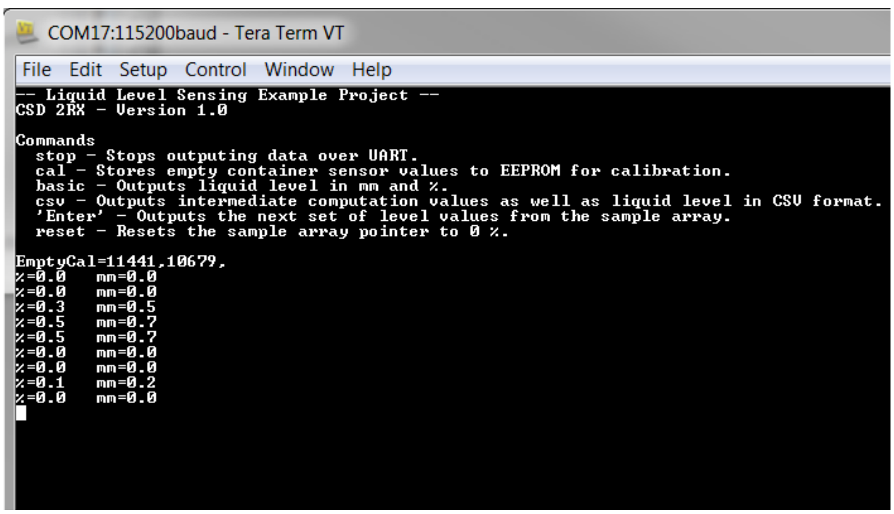
   Open the Tera Term Serial port setup dialog at **Setup** > **Serial port…** Only the **Baud rate:** should require changing to 115200 but it is good to also confirm the other settings as shown in Figure 5.

Figure 5. Terminal Emulator Setup Parameters



3. Serial output is displayed in the terminal window as shown in Figure 6. On startup, the PSoC device will output information on the project name, version, supported commands, empty sensor calibration values, and continuous output of the current liquid level in millimeters and percent.

Figure 6. Startup Terminal Window Output



Components

Table 1 lists the PSoC Creator Components used in this example, as well as the hardware resources used by each.

Table 1. List of PSoC Creator Components

Component	Name	Hardware Resources	Non-default Parameter Settings
CapSense CSD [v2.30]	CapSense_CSD	1 CapSense block 2 or 12 GPIO pins for sensors 1 GPIO pin for Cmod capacitor	General Tuning method: Manual with run time tuning Widgets Generics: 2 or 12 Scan resolution: 14 bits Scan Order Modulation IDAC: Not critical as this value is set by firmware Compensation IDAC: 0 Advanced IDAC range: 8x

Component	Name	Hardware Resources	Non-default Parameter Settings
			Analog switch: PRS-12b Individual freq settings: Disabled Sense clock divider: 10 Modulator clock divider: 10
UART (SCB mode) [v3.0]	UART	1 SCB block	RX buffer size: 32 TX buffer size: 32
Emulated EEPROM [v1.10]	Em_EEPROM	1 Flash block	None
Digital Output Pin [v2.10]	Pin_TxGnd	1 GPIO pin for 2RX sensor only	Drive mode: Strong Initial drive state: 0

Design-Wide Resources

Figure 7 shows the pin selections for the 2RX project and Figure 8 shows the pin selections for the 12RX project.

Figure 7. 2RX project **Pins** Tab in Design Wide Resources (**.cydwr** file)

Name	Port	Pin	Lock
\CapSense_CSD:Cmod\ (Cmod)	P4[2]	22	✓
\CapSense_CSD:Sns[0]\ (Generic0_0__GEN)	P0[0]	24	✓
\CapSense_CSD:Sns[1]\ (Generic1_0__GEN)	P3[7]	18	✓
\UART:rx\	P0[4]	28	✓
\UART:tx\	P0[5]	29	✓
Pin_TxGnd	P2[5]	7	✓

Figure 8. 12RX project **Pins** Tab in Design Wide Resources (**.cydwr** file)

Name	Port	Pin	Lock
\CapSense_CSD:Cmod\ (Cmod)	P4[2]	22	✓
\CapSense_CSD:Sns[0]\ (Generic0_0__GEN)	P2[4]	6	✓
\CapSense_CSD:Sns[1]\ (Generic1_0__GEN)	P2[3]	5	✓
\CapSense_CSD:Sns[2]\ (Generic2_0__GEN)	P2[2]	4	✓
\CapSense_CSD:Sns[3]\ (Generic3_0__GEN)	P1[3]	40	✓
\CapSense_CSD:Sns[4]\ (Generic4_0__GEN)	P1[5]	42	✓
\CapSense_CSD:Sns[5]\ (Generic5_0__GEN)	P3[6]	17	✓
\CapSense_CSD:Sns[6]\ (Generic6_0__GEN)	P2[6]	8	✓
\CapSense_CSD:Sns[7]\ (Generic7_0__GEN)	P2[7]	9	✓
\CapSense_CSD:Sns[8]\ (Generic8_0__GEN)	P1[0]	37	✓
\CapSense_CSD:Sns[9]\ (Generic9_0__GEN)	P3[5]	16	✓
\CapSense_CSD:Sns[10]\ (Generic10_0__GEN)	P0[0]	24	✓
\CapSense_CSD:Sns[11]\ (Generic11_0__GEN)	P3[7]	18	✓
\UART:rx\	P0[4]	28	✓
\UART:tx\	P0[5]	29	✓

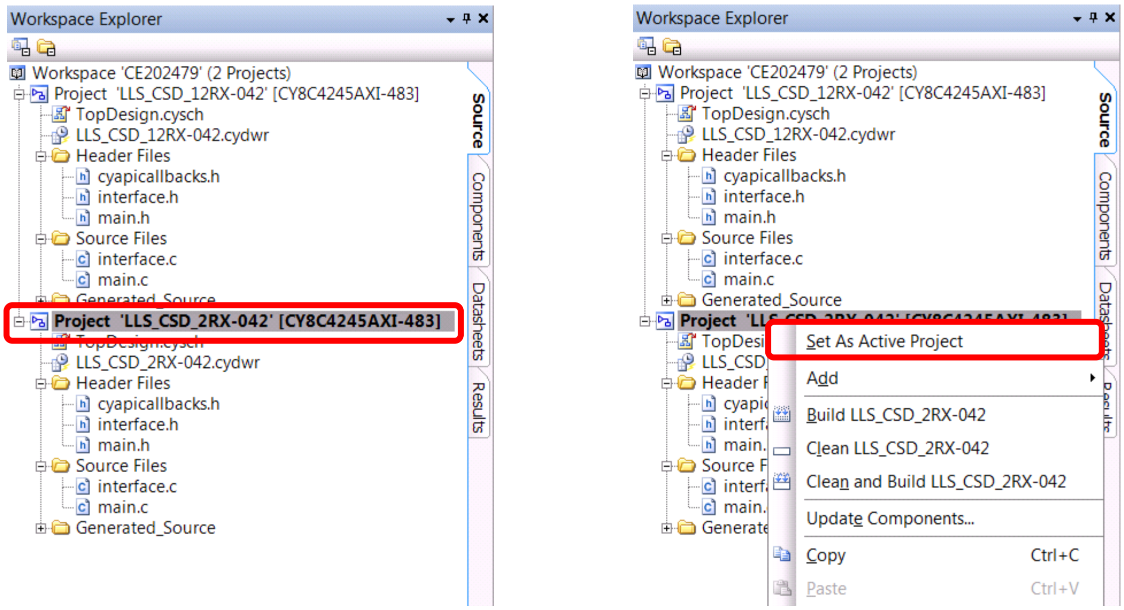
Basic Operation

1. Load the workspace into PSoC Creator by opening

<Download Directory>\CE202479\CE202479.cywrk.

2. Select the project you wish to program into the PSoC device and set it as the active project by right-clicking on the project branch and selecting ‘Set As Active Project’ as shown in Figure 9. The active project name will be displayed in bold.

Figure 9. Active Project

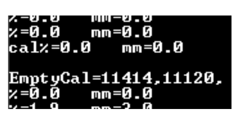


3. Build the example project by navigating to **Build** > **Build *<Project Name>***.
4. The default programming interface for the kit is a USB-based, onboard programming interface. To program the device, plug the USB cable into the USB connector of the CY8CKIT-042 Pioneer kit. The kit will enumerate as a composite device. For more details about programming, refer to the Programming and Debugging section of your Pioneer kit’s user guide.
5. Program the example to the device by choosing **Debug** > **Program**.
6. Power the device, if not already powered.
7. Calibrate sensors for an empty container on first use to compensate for system differences and manufacturing tolerances that effect parasitic capacitance. Calibration stores the empty sensor value for each sensor in flash memory and automatically reloads them on future device startups. **(see Errata Information to correct calibration reload error)** The empty calibration values are subtracted from all future sensor values to provide a consistent reference value for use in calculations. Any changes to CapSense tuning or the sensor connection will require updated calibration values. Execute the calibrate command with just enough liquid in the container to reach the 0mm (0%) mark on the sensor.

   Note: Reprogramming the PSoC device will overwrite previously stored calibration values.

   a. Terminal – Type ‘cal’ into the terminal. Calibration values for each sensor will be displayed and output will return to the previously selected mode as shown in Figure 10.

Figure 10. Terminal Calibration



8. Fill and empty the container as desired with water and view the calculated liquid level.

   a. The default output displays the current liquid level in percentage values and millimeters to one decimal place approximately once every second. See Figure 6 for an example. Entering the command `basic` at any time returns to the default output.

Water that is significantly colder than the ambient air temperature may cause condensation to form on the sensor surface depending on the humidity level. Condensation may cause the liquid level calculation to return a liquid level with increased error. Condensation during low-temperature evaluation can be reduced by insulating the exposed surface of the sensor.

Hot water above 170°F / 80°C can cause PET plastic to deform. Extremely hot water should be avoided as the liquid container provided in the CY8CKIT-022 kit is made of PET plastic.

Variations in the hardware setup can change the sensor capacitance and increase error in the reported level. Common variations include touching the sensor during operation and moving the liquid container or Pioneer board after calibration.

Advanced Operation

A set of commands is provided to assist accuracy measurement and analysis of the liquid level system. An array of heights matching the millimeter ruler printed on the CY8CKIT-022 sensors is provided in firmware. The predefined levels are -5, 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 153, and 160 mm. While filling the liquid container, the **Output Value** command can be executed when the predefined level is reached. On execution of the command, the predefined level value, sensor difference values, and the currently calculated level are output. The predefined level is automatically incremented in preparation for the next **Output Value** command. This process simplifies taking a complete set of accuracy measurements for analysis because it documents both the expected and actual level together. A second command, **Reset Array,** allows resetting the predefined level to its initial value to start a new test cycle. A full test cycle is shown in Figure 11. .

Figure 11. Accuracy Measurement Example

```
reset
Reset Test Level
Presetmm,SenDiff0,SenDiff1,Level%,Levelmm
-5,101,1,0.9,1.5
0,109,1,0.9,1.3
10,272,20,7.3,11.2
20,397,51,12.8,19.7
30,492,84,17.1,26.2
40,588,132,22.5,34.4
50,676,197,29.2,44.7
60,760,269,35.5,54.3
70,839,358,42.8,65.5
80,914,451,49.5,75.7
90,977,548,56.3,86.1
100,1037,648,62.7,95.9
110,1091,753,69.2,106.0
120,1136,859,75.9,116.1
130,1171,949,81.3,124.4
140,1204,1061,88.4,135.3
150,1222,1148,94.3,144.2
153,1230,1178,96.1,147.0
160,1231,1186,96.7,147.9
```

- Pressing the `[Enter]` key with no other characters executes the **Output Value** command. Repeated command execution results in an ordered table of liquid levels suitable for data logging. Entering the command `reset` executes the **Reset Array** command.

- Executing the terminal `stop` command causes the continuous output to halt. This is useful to ensure only desired data is logged. Execution of any other command will resume the output.
- Executing the terminal `csv` command causes internal sensor values as well as the liquid level to be output in the comma-separated value format for easy import into spreadsheet programs. The first line output after the `csv` command is entered is the header with a label describing what each value is.

Errata Information

A bug in the firmware has been discovered that causes the calibration values to return '0' after a reset event. Please make the following edits in project LLS_CSD_12RX-042. No changes are required for project LLS_CSD_2RX-042.

1. In *main.c*, line 79, replace the code with:

   ```
   sensorEmptyOffset[i] = ((volotile int16 *))eepromEmptyOffset)[i];
   ```

2. In *main.c,* line 56, replace the code with:

   ```
   const int16 CYCODE eepromEmptyOffset[NUMSENSORS] = {0u};
   ```

3. In *interface.c,* line 92, replace the code with:

   ```
   status = Em_EEPROM_Write((uint8 *)sensorEmptyOffset, (uint8 *)eepromEmptyOffset,
   sizeof *sensorEmptyOffset * NUMSENSORS);
   ```

Related Documents

Table 2 lists all relevant application notes, code examples, knowledge base articles, device datasheets, and Component datasheets.

Table 2. Related Documents

Application Notes		
AN202478	AN202478 – PSoC® 4 - Capacitive Liquid Level Sensing	Provides design, sensor layout, and tuning guidance for liquid level sensing with PSoC 4 devices
AN85951	AN85951 – PSoC 4® CapSense Design Guide	Shows how to design capacitive touch sensing applications with the PSoC 4 and PRoC BLE families of devices and use the CapSense component.
Code Examples		
CE195286	CapSense CSD using Tuner with PSoC 4	
CE95288	CapSense Low Power with PSoC 4	
PSoC Creator Component Datasheets		
CapSense CSD	Capacitive sensing using a Delta-Sigma Modulator	
UART (SCB mode)	Provides asynchronous communications commonly referred to as RS232 or RS485.	
Digital Output Pin	The Pins component allows hardware resources to connect to a physical port-pin	
Device Documentation		
PSoC 4 Datasheets		
PSoC 4 Technical Reference Manuals		
Development Kits		
PSoC 4 Kits		
Tool Documentation		
PSoC™ Creator user guide		
PSoC™ Creator quick start guide		

Document History

Document Title: CE202479 - PSoC™ 4 Capacitive Liquid Level Sensing

Document Number: 002-02479

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5032590	GJV	12/02/2015	New code example
*A	5730193	AESATP12	05/16/2017	Updated logo and copyright.
*B	6484923	GJV	02/15/2019	Added errata to correct firmware error causing calibration data to return ‘0’ after reset.
*C	8046116	SASH	07/15/2024	Removed Registered symbol from the title. Removed content related to uProbe. Updated weblinks

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

Technical Support

cypress.com/support

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