

## AN2397

**PSoC<sup>®</sup> 1 and CapSense<sup>®</sup> Controllers – CapSense Data Monitoring Tools**

Author: Kurian Polachan

Associated Project: No

Associated Part Family: CY8CMBR3xxx, CY8CMBR2xxx,  
CY8C201xx, CY8C20x34, CY8C20xx6A/AS/H, CY8C20xx7/S,  
CY8C21x34/B, CY8C24x94, CY8C22x45

Software Version: Bridge Control Panel 1.12

Related Application Notes: [AN49943](#), [AN42137](#)

AN2397 shows how to monitor CapSense data from PSoC 1 and CapSense controllers using the I<sup>2</sup>C or UART interface. With the tools described in this application note, you can view and log real-time sensor data for CapSense tuning and debugging.

**Contents**

1	Introduction.....	1	5.3	I <sup>2</sup> C with EZ-Click for CY8CMBR2110 and CY8CMBR3xxx Controllers.....	14
2	CapSense Resources.....	2	5.4	UART with BCP for CY8CMBR20xx Controllers.....	18
2.1	PSoC Designer.....	2	6	Enabling UART-to-USB Bridge Using CY3240-I2USB Bridge.....	21
2.2	Code Examples.....	3	6.1	Converting CY3240-I2USB to UART-to-USB Bridge.....	21
2.3	Technical Support.....	4	6.2	UART-to-USB Driver Installation.....	21
3	Selecting the Right Method.....	4	7	Glossary.....	24
4	Setup.....	5	8	Summary.....	28
4.1	Monitoring via I <sup>2</sup> C.....	6		Worldwide Sales and Design Support.....	30
4.2	Monitoring via UART.....	6			
5	Methods for CapSense Data Monitoring.....	6			
5.1	I <sup>2</sup> C with BCP for Programmable Controllers.....	6			
5.2	UART with BCP for Programmable Controllers 10				

**1 Introduction**

During the CapSense design process, you will need to monitor CapSense sensor data, such as raw count, baseline, and difference count, for tuning and debugging.

This document helps you to select the proper tool for viewing and logging CapSense sensor data. The two supported communication interfaces are I<sup>2</sup>C and UART. You should be familiar with CapSense sensing technology before you read this document. If you would like more information about general CapSense theory and operation, see [Getting Started with CapSense](#).

Depending on the device type you are using for design, use one of the following resources:

- **PSoC 1 and CapSense Controllers:** Refer to the [Selecting the Right Method](#) section in this application note.
- **PSoC 3 or PSoC 5LP:** Refer to the “Tuner GUI” section in the [PSoC 3 and PSoC 5LP CapSense Design Guide](#).
- **PSoC 4:** Refer to the “Manual Tuning Process” section in the [PSoC 4 CapSense Design Guide](#).

## 2 CapSense Resources

Cypress provides a wealth of data at [Cypress CapSense Controllers website](#) to help you to select the right CapSense device for your design, and quickly and effectively integrate the device into your design. To learn more about CapSense and the available design resources, refer to the application note [AN64846 – Getting Started with CapSense](#).

The following is an abbreviated list for CapSense:

- **Overview:** CapSense Portfolio – Refer to the *CapSense Product Portfolio* section in [AN64846, CapSense Roadmap](#).
- **Product Selectors:** Refer to the *CapSense Selector Guide* section in [AN64846](#). In addition, [PSoC Designer](#) includes a device selection tool.
- **Datasheets** describe and provide electrical specifications for the CapSense device family.
- **Application Notes and Code Examples** cover a broad range of topics, from basic to advanced level. Many of the application notes include code examples.
- **Technical Reference Manuals (TRM)** provide detailed descriptions of the internal architecture of the CapSense devices.
- **Development Kits:**
  - [CY3280-MBR3 CapSense MBR3 Evaluation Kit](#) is designed to showcase the abilities of the CapSense MBR3 devices. MBR3 devices are register-configurable devices and support buttons, proximity sensing, water tolerance and many more features.
  - [CY3280-BK1 Universal CapSense Controller - Basic Kit 1](#) enables you to evaluate the programmable CapSense controllers. This kit contains the evaluation boards for CY8C20x34 and CY8C21x34 devices.
  - [CY8CKIT-024 CapSense Proximity Shield](#) enables you to evaluate and develop CapSense proximity applications. The shield has been designed to work with any of the Cypress [Pioneer development platforms](#).
- The [MiniProg1](#) and [MiniProg3](#) devices provide an interface for flash programming.

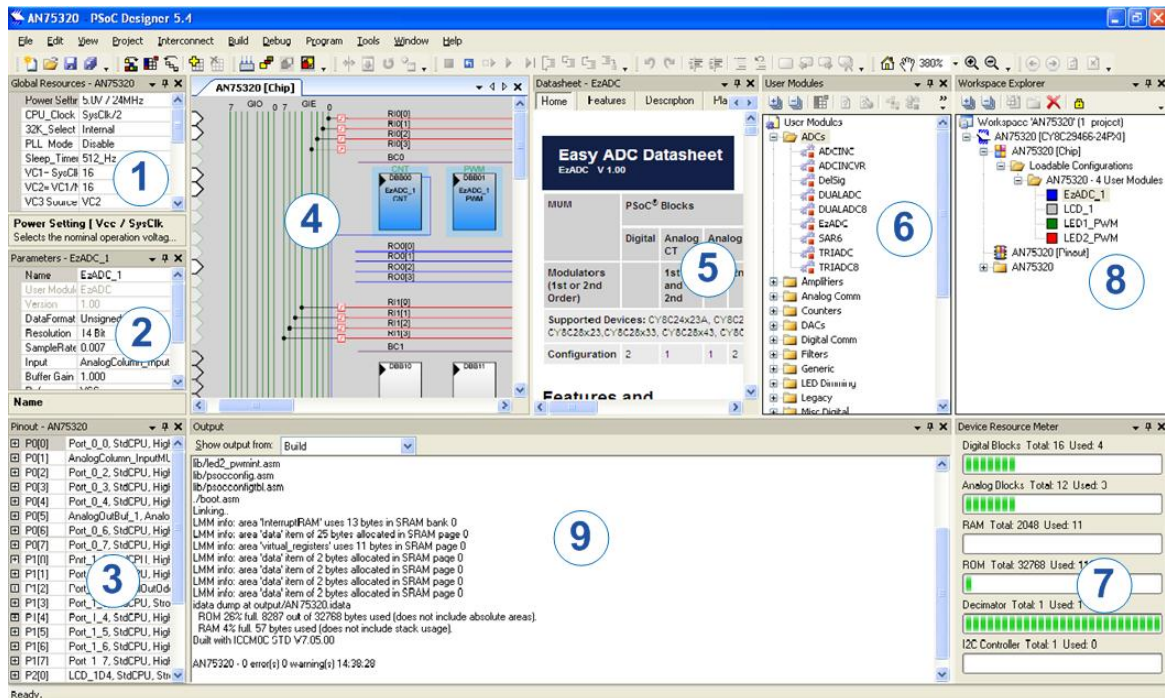
### 2.1 PSoC Designer

[PSoC Designer](#) is a free Windows-based Integrated Design Environment (IDE). Develop your applications using a library of precharacterized analog and digital peripherals in a drag-and-drop design environment. Then, customize your design leveraging the dynamically generated API libraries of code. [Figure 1](#) shows PSoC Designer windows.  
**Note:** This is not the default view.

1. **Global Resources** – all device hardware settings.
2. **Parameters** – the parameters of the currently selected User Modules
3. **Pinout** – information related to device pins
4. **Chip-Level Editor** – a diagram of the resources available on the selected chip
5. **Datasheet** – the datasheet for the currently selected UM
6. **User Modules** – all available User Modules for the selected device
7. **Device Resource Meter** – device resource usage for the current project configuration
8. **Workspace** – a tree-level diagram of files associated with the project
9. **Output** – output from project build and debug operations

**Note:** For detailed information on PSoC Designer, go to **PSoC® Designer > Help > Documentation > Designer Specific Documents > IDE User Guide**.

Figure 1. PSoC Designer Layout



## 2.2 Code Examples

This [webpage](#) lists the PSoC Designer based Code Examples. These Code Examples can speed up your design process by starting you off with a complete design, instead of a blank page, and show how PSoC Designer User Modules can be used for various applications. In addition, the [CapSense Controller Code Examples](#) design guide provides important code examples to be used in a CapSense design.

To access the Code Examples integrated with PSoC Designer, follow the path **Start Page > Design Catalog > Launch Example Browser** as shown in [Figure 2](#).

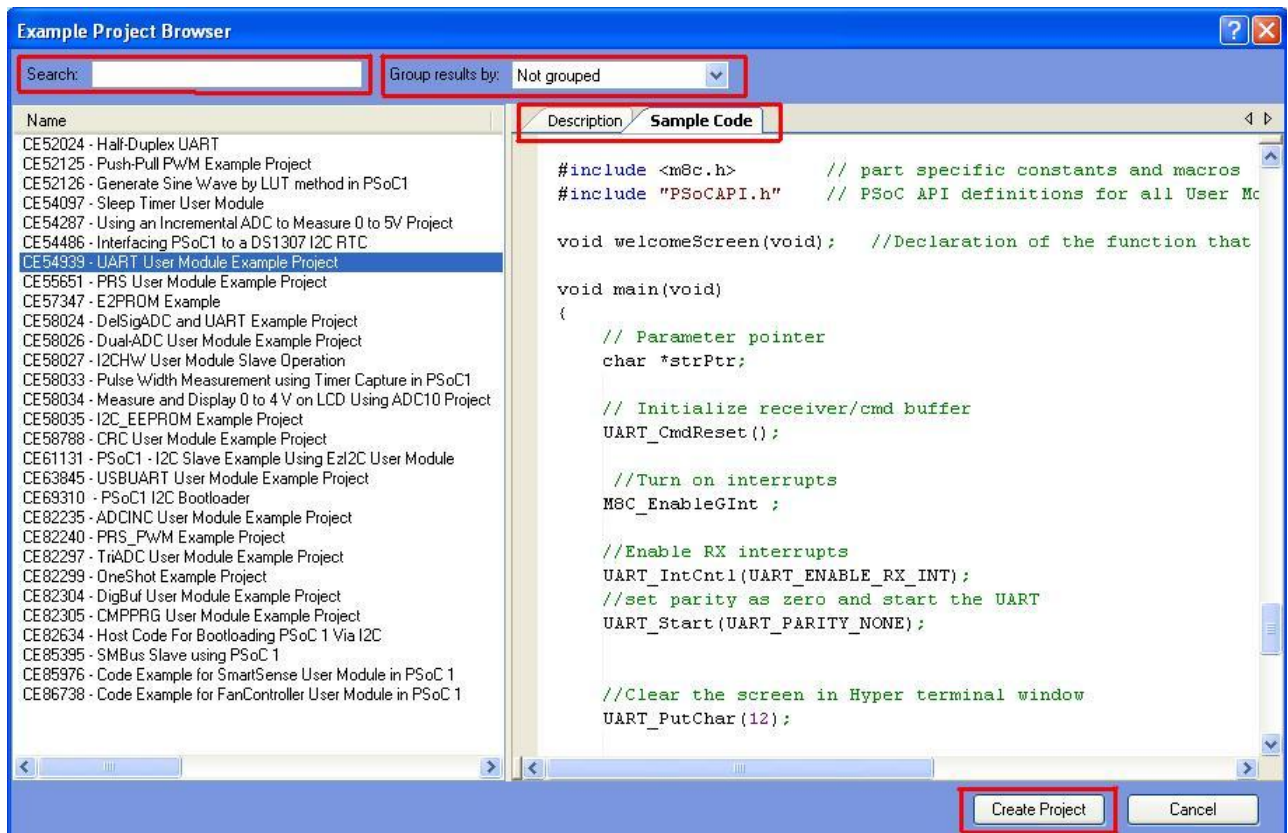
Figure 2. Code Examples in PSoC Designer



In the Example Projects Browser shown in [Figure 3](#), you have the following options.

- Keyword search to filter the projects
- Listing the projects based on Category
- Review the datasheet for the selection (on the Description tab)
- Review the code example for the selection. You can copy and paste code from this window to your project, which can help speed up code development, or
- Create a new project (and a new workspace if needed) based on the selection. This can speed up your design process by starting you off with a complete, basic design. You can then adapt that design to your application.

Figure 3. Code Example Projects, with Sample Codes



## 2.3 Technical Support

If you have any questions, our technical support team is happy to assist you. You can create a support request on the [Cypress Technical Support page](#).

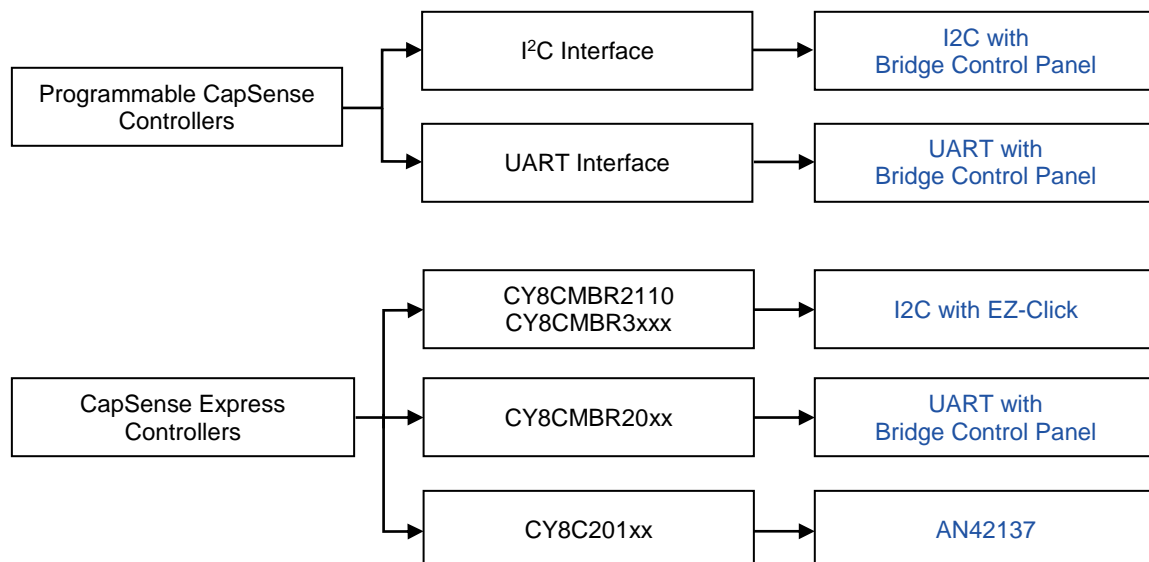
You can also use the following support resources if you need quick assistance.

- [Self-help](#)
- [Local Sales Office Locations](#)

## 3 Selecting the Right Method

Cypress offers a wide range of programmable and configurable (CapSense Express) CapSense controllers. The communication interface and the tool used for data monitoring differ depending on the device. [Figure 4](#) guides which method to select. Click the link to go to the corresponding section. The hardware setup is presented in the next section.

Figure 4. Data Monitoring Method Selection Chart



For the CY8C201xx family, the CapSense data viewing tool is integrated directly into [PSoC Designer™ version 5.0 SP6](#). Refer to [AN42137](#) for more information.

For the programmable CapSense controllers, either I²C or UART can be used as communication interface. [Table 1](#) guides which interface to use.

Table 1. I2C vs. UART for Programmable Controllers

Comparison Matrices	I²C	UART
Pins Required	2	1
Data Loss	Yes	No
Bidirectional Communication Support	Yes	No

**Note** EZ-Click does not support reading sensor data from programmable controllers.

- I²C requires two free port pins (SCL and SDA) from the CapSense controller and UART needs only one pin, the transmitter pin of the TX8 user module.
- In the I²C method, the PC reads data from the CapSense controller asynchronously (asynchronous to CapSense scanning). In this case, samples will be lost if the CapSense scanning rate is higher than the rate at which the PC reads data from the CapSense controller. In the UART method, data from the CapSense controller is synchronously transmitted to the PC (data is transmitted after every CapSense scan). No samples are lost in this mode of communication.
- BCP does not support sending data to the CapSense controller over UART.

## 4 Setup

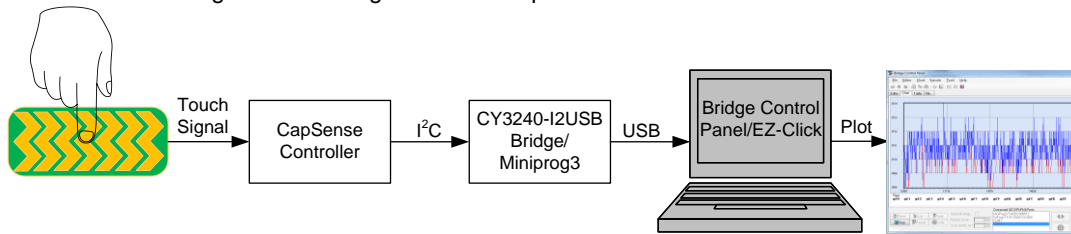
CapSense controllers support either I²C or UART interface to monitor CapSense data. The hardware setup differs based on the communication interface. Choose the right setup depending on the interface supported by your CapSense device.



## 4.1 Monitoring via I<sup>2</sup>C

In this method, data is read from the CapSense controller via the I<sup>2</sup>C interface and is transmitted to the PC through USB using the [CY3240-I2USB<sup>a</sup>](#) Bridge (I<sup>2</sup>C-to-USB converter) or the [MiniProg3](#) kit, as shown in [Figure 5](#). On the PC side, you use the [Bridge Control Panel](#) (BCP) or [EZ-Click™](#) tool to view and log the sensor data. BCP is a tool provided by Cypress to read the data over I<sup>2</sup>C or UART and is installed along with [PSoC Programmer](#). EZ-Click is used to set up the sensor configuration, monitor the real-time sensor output, and run production-line system diagnostics for [CY8CMBR2110](#) and [CY8CMBR3xxx](#) devices.

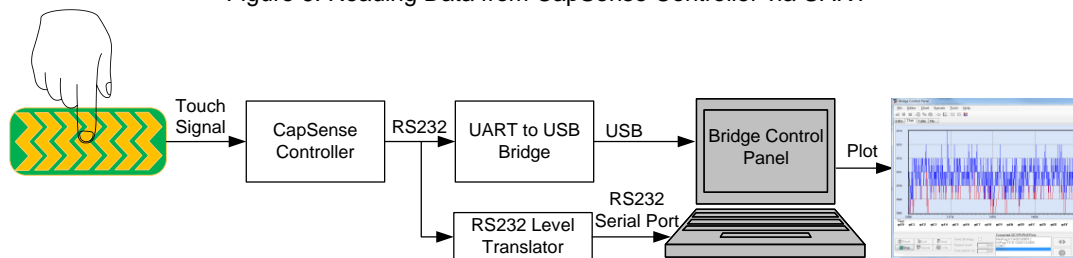
Figure 5. Reading Data from CapSense Controller via I<sup>2</sup>C Interface



## 4.2 Monitoring via UART

In this method, data is read from the CapSense controller via a UART/RS232 interface and is transmitted to the PC through the USB interface or RS232 serial port, as shown in [Figure 6](#). On the PC side, you need to use the [BCP](#) tool to view and log the sensor data.

Figure 6. Reading Data from CapSense Controller via UART



RS232 data transmitted by the CapSense controller can be sent to the PC using one of the following methods:

- UART-to-USB bridge: Refer to the section [Enabling UART-to-USB Bridge Using CY3240-I2USB Bridge](#) to use the [CY3240-I2USB](#) kit. Refer to [AN49943](#) for implementing a UART-to-USB Bridge for a PSoC 1 device.
- RS232 level translator: If your PC has an RS232 serial port, you can use an RS232 level translator to transmit the data from the CapSense controller to the PC.

# 5 Methods for CapSense Data Monitoring

## 5.1 I<sup>2</sup>C with BCP for Programmable Controllers

Code example 2 in the [CapSense Controller Code Examples Design Guide](#) demonstrates how to read CapSense data from an I<sup>2</sup>C slave device and plot it in the BCP tool.

### 5.1.1 Step A: Install BCP

1. The BCP tool installs with PSoC Programmer. Download it from [www.cypress.com/programmer](http://www.cypress.com/programmer).

<sup>a</sup> [CY3240-I2USB](#) kit is obsolete and [MiniProg3](#) is the replacement. However, CY3240-I2USB kit is provided along with other CapSense kits – [CY3280-BK1](#), [CY3280-20x66](#), [CY3218-CAEXP1](#), [CY3218-CAEXP2](#).

2. After installation, open the application by going to **Start > All Programs > Cypress > Bridge Control Panel [version] > Bridge Control Panel [version]**.

### 5.1.2 Step B: Implement I<sup>2</sup>C Slave Interface in CapSense Controller

1. Create a new project for any one of the programmable CapSense controllers in PSoC Designer.
2. Place an EzI2Cs User Module in the project.
3. Set the following User Module parameters:
  - Slave\_Addr to 0 (this address can be anything from 0 to 127)
  - Address\_Type to Static
  - ROM\_Registers, in most cases, to Disable
  - I2C Clock to 400 kHz Fast
  - I2C Pins to P1[0]-P1[1] or P1[5]-P1[7]
4. Define the RAM buffer that will contain the data needed for I<sup>2</sup>C transmission. Refer to the [code example 2](#) to learn how to do this. Define the array or structure whose address will be specified when the SetRamBuffer function is called. For example:

```
struct I2C_Regs
{
    BYTE bSnsIndex;    // read/write value
    BYTE bSnsMask;     // read only value
    WORD wRawCount;    // read only value
    WORD wBaseline;    // read only value
    WORD wDiffCount;   // read only value
    WORD wCentroid;    // read only value
} MyI2C_Regs;
```

5. Insert the following two lines into the initialization part of the program:

```
EzI2Cs_SetRamBuffer(sizeof(MyI2C_Regs), 1, (BYTE*)&MyI2C_Regs);

EzI2Cs_Start();
```

Modify the first line to set the arguments of the EzI2Cs\_SetRamBuffer function as required. The first argument sets the length of the data that can be read. The second sets the length of the data that can be written to. The third argument sets the address of the I<sup>2</sup>C buffer. Refer to [EzI2C Slave User Module Datasheet](#) for details on these functions.

6. Update the RAM buffer after every scan. For example:

```
MyI2C_Regs.bSnsMask = CSD_baSnsOnMask[0];
MyI2C_Regs.wRawCount = CSD_waSnsResult[MyI2C_Regs.bSnsIndex];
MyI2C_Regs.wBaseline = CSD_waSnsBaseline[MyI2C_Regs.bSnsIndex];
MyI2C_Regs.wDiffCount = CSD_waSnsDiff[MyI2C_Regs.bSnsIndex];
MyI2C_Regs.wCentroid = CSD_wGetCentroidPos(1);
```

The EzI2Cs User Module works in the background with I<sup>2</sup>C interrupt handlers.

The first byte received by the I<sup>2</sup>C slave is the offset into the buffer, which data is read from or written to. The default offset is 0. To learn more about working with the EzI2Cs User Module, refer to the [user module datasheet](#).

**Note** To guarantee data integrity of variables that are two or more bytes long, check the EzI2Cs\_bBusy\_Flag variable before changing these variables in the code. For example:

```
EzI2Cs_DisableInt();
if(!(EzI2Cs_bBusy_Flag == EzI2Cs_I2C_BUSY_RAM_READ))
{
    MyI2C_Regs.wData++; //safely increment a 2 byte variable
}

EzI2Cs_ResumeInt();
```

### 5.1.3 Step C: Set Up the Hardware

Figure 5 shows how to interconnect the components for this method.

If you are using the CY3240-I2USB Bridge, see the [CY3240-I2USB Bridge Guide](#) for information on how to connect the I<sup>2</sup>C pins of the CapSense controller to the CY3240-I2USB Bridge.

If you are using MiniProg3, see the [MiniProg3 User Guide](#) for information on how to connect the I<sup>2</sup>C pins of the CapSense controller to the MiniProg3.

### 5.1.4 Step D: Read CapSense Sensor Data Using BCP

Figure 7, Figure 8, and Figure 9 show screenshots of various BCP windows. Help topics are available in the BCP menu bar for additional information about the application user interface, supported commands, variable definition, and different modes of data acquisition (Repeat Mode, To File Mode).

Figure 7. BCP: Commands Editor View

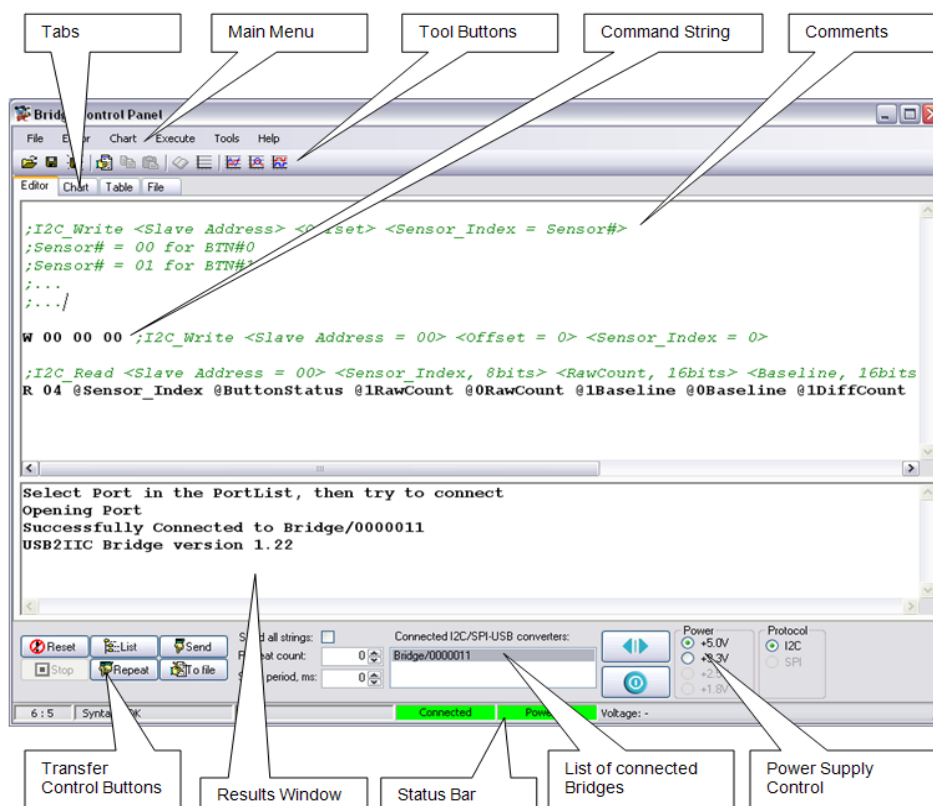




Figure 8. BCP: Chart View (Plotting of RawCount and Baseline)

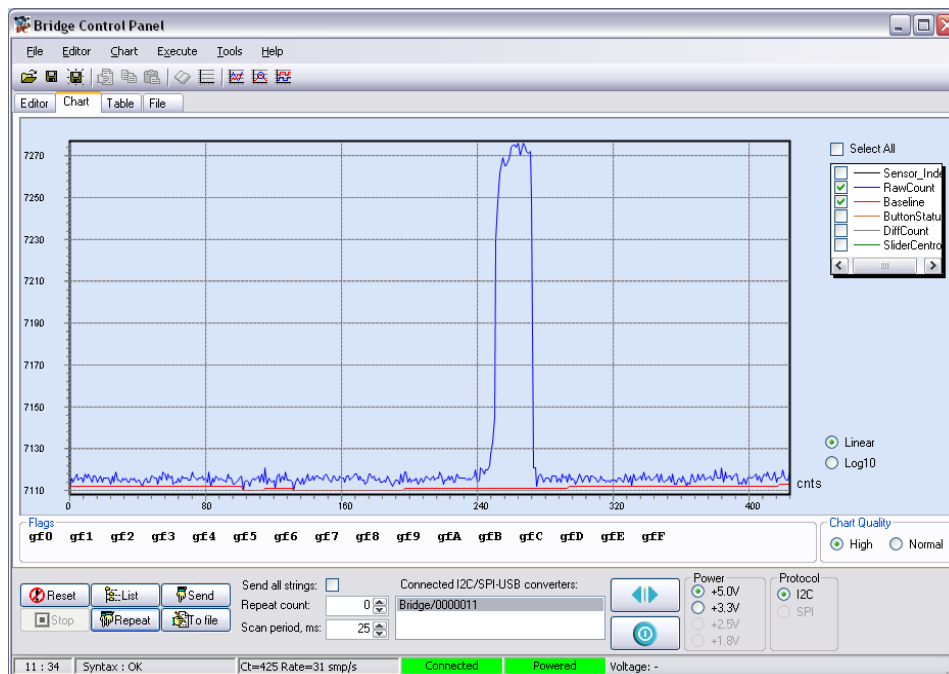
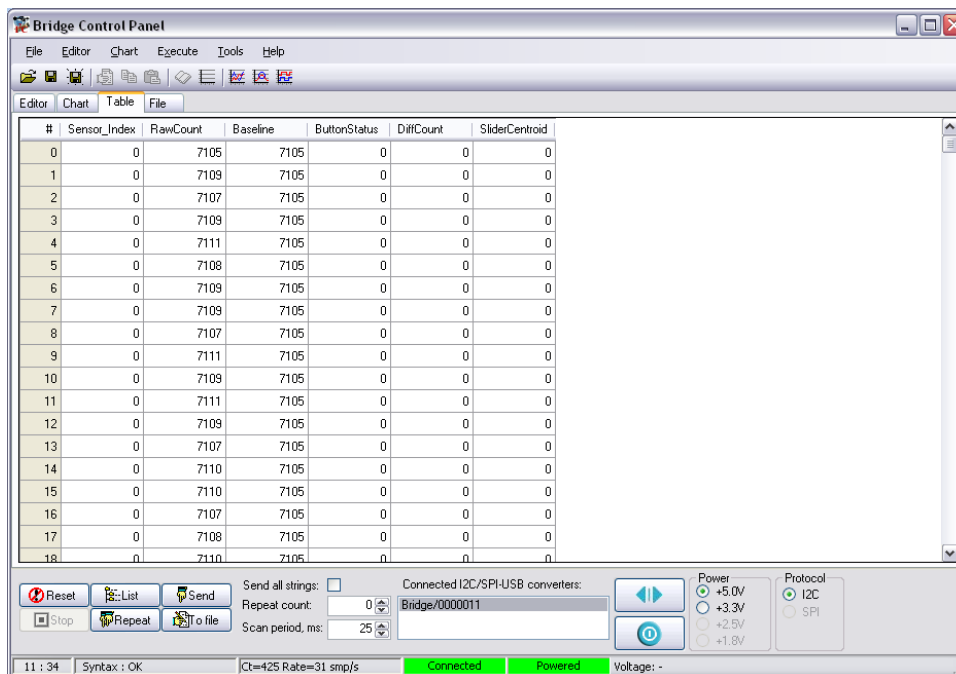


Figure 9. BCP: Table View of Sensor Data



The Bridge Control Panel (BCP) Table View displays sensor data in a table format. The table has the following columns: #, Sensor\_Index, RawCount, Baseline, ButtonStatus, DiffCount, and SliderCentroid. The data is organized into a table with 19 rows. The status bar at the bottom indicates the connection status (Connected, Powered) and voltage levels.

#	Sensor_Index	RawCount	Baseline	ButtonStatus	DiffCount	SliderCentroid
0	0	7105	7105	0	0	0
1	0	7109	7105	0	0	0
2	0	7107	7105	0	0	0
3	0	7109	7105	0	0	0
4	0	7111	7105	0	0	0
5	0	7108	7105	0	0	0
6	0	7109	7105	0	0	0
7	0	7109	7105	0	0	0
8	0	7107	7105	0	0	0
9	0	7111	7105	0	0	0
10	0	7109	7105	0	0	0
11	0	7111	7105	0	0	0
12	0	7109	7105	0	0	0
13	0	7107	7105	0	0	0
14	0	7110	7105	0	0	0
15	0	7110	7105	0	0	0
16	0	7107	7105	0	0	0
17	0	7108	7105	0	0	0
18	0	7110	7105	0	0	0

## 5.2 UART with BCP<sup>a</sup> for Programmable Controllers

### 5.2.1 System Requirements

This method requires the following system configuration:

- A free serial communication (COM/RS232) port <sup>[b]</sup>
- Microsoft Windows XP, 7, or later

Refer to the section [Enabling UART-to-USB Bridge Using CY3240-I2USB Bridge](#) to use CY3240-I2USB kit. Refer to [AN49943](#) for implementing UART-to-USB Bridge for a PSoC 1 device.

Follow these steps to view CapSense data using this method.

### 5.2.2 Step A: Install BCP

The BCP tool installs with PSoC Programmer. Download it from [www.cypress.com/programmer](http://www.cypress.com/programmer).

### 5.2.3 Step B: Send Data from CapSense Controller

For programmable devices, follow these steps in a PSoC Designer Project:

1. Place the TX8SW User Module in the PSoC Designer project.
2. Set the following User Module parameters:
  - **Port Pin:** Select the RS232 serial transmitter pin.
  - **Baud Rate:** Select a baud rate from the list. Possible choices are 115200, 57600, 38400, 19200, 9600, 4800, 2400, and 1200 bits per second.
  - **Parity:** Set to NONE.
  - **Stop Bits:** Set to ONE.
  - **Data Bits:** Set to 8.
3. Insert the following string into the initialization part of the program:
 

```
TX8SW_Start();
```
4. Now we will learn how to send CapSense data in a specific packet structure so that BCP can parse the data. The code snippets given here should be inserted in the main loop after CapSense scan is done. The BCP application receives the data that is divided into packets:

Packet0	Packet1	...	PacketN	...
---------	---------	-----	---------	-----

Each packet consists of a header, a data element, and a tail, as described in [Table 2](#). The RX8 command entered in the BCP application should match with the packet structure sent by PSoC, as explained in [Step D: Read CapSense Sensor Data Using BCP](#), to correctly parse the packet. We will assume the packet format given in [Table 2](#) and write the code accordingly.

<sup>a</sup> Multichart used for data monitoring over UART is replaced by Bridge Control Panel and is no longer supported.

<sup>b</sup> This requirement is optional if an UART-to-USB bridge is used for reading serial (RS232) data.

Table 2. Data Packet Structure

Packet Element	Content
Header	0x0D 0x0A
Data	Raw Count 0 Raw Count 1 ... Baseline 0 Baseline 1 ... Signal 0 Signal 1 ...
Tail	0x00 0xFF 0xFF

5. The header allows the BCP to divide the input byte stream into packets. It can contain any sequence of bytes that is not part of the data. The header used here consists of two bytes: a carriage return (0x0D) and a line feed (0x0A). To send the header, call the standard TX8 User Module function:

```
TX8SW_PutCRLF(); // Send Header
```

The header is followed by data. The data section consists of three 16-bit words (Rawcount, Baseline, Diffcount) for each sensor. Each word is represented using one unsigned integer variable<sup>a</sup> in BCP. Data words should be ordered according to [Table 2](#). The following code sequence sends the data section, with all sensor results in the proper order, to the PC:

```
TX8SW_Write((char *) (CSD_waSnsResult), CSD_TotalSensorCount*2);  
TX8SW_Write((char *) (CSD_waSnsBaseline), CSD_TotalSensorCount*2);  
TX8SW_Write((char *) (CSD_waSnsDiff), CSD_TotalSensorCount*2);
```

For 16-bit values like RawCount, the MSB is sent out first, followed by the LSB.

6. The packet is bounded by the tail, which allows BCP to determine the packet length. Again, the tail can contain any sequence of bytes that is not part of data. The tail used here consists of three bytes: 0x00 0xFF 0xFF. To send the tail, add the following code to your project:

```
TX8SW_PutChar(0x00); // Send Tail  
TX8SW_PutChar((CHAR) 0xFF);  
TX8SW_PutChar((CHAR) 0xFF);
```

#### 5.2.4 Step C: Set up the hardware

[Figure 6](#) shows how to interconnect the components for this method. Follow these rules if the CY3240-I2USB Bridge is used as a UART-to-USB bridge.

- Connect the serial transmitter pin of the CapSense controller to the I2C\_SDA pin of the CY3240-I2USB Bridge.
- Wire the ground of the CapSense controller to the GND pin of the CY3240-I2USB Bridge.

See the [CY3240-I2USB Bridge](#) user guide for details about the CY3240-I2USB pinouts.

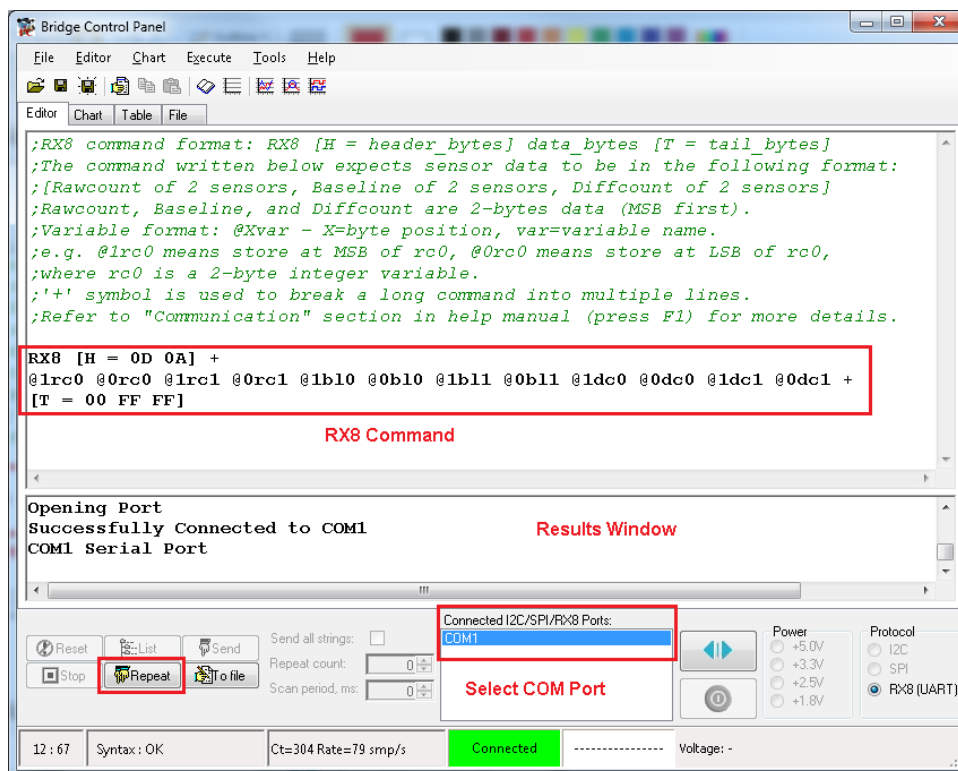
<sup>a</sup> Each data is represented using a named variable in BCP. e.g. rc0 can be used to represent the rawcount of sensor 0. BCP supports up to 32 such variables. The data type of variables can be byte, int, long int, and float and they can be either signed or unsigned. Refer to BCP help manual for more information.

### 5.2.5 Step D: Read CapSense Sensor Data Using BCP

To read the CapSense sensor data using BCP, follow these steps:

1. Open the BCP application by going to **Start > All Programs > Cypress > Bridge Control Panel [version] > Bridge Control Panel [version]**. Figure 10 shows the main window of the application.

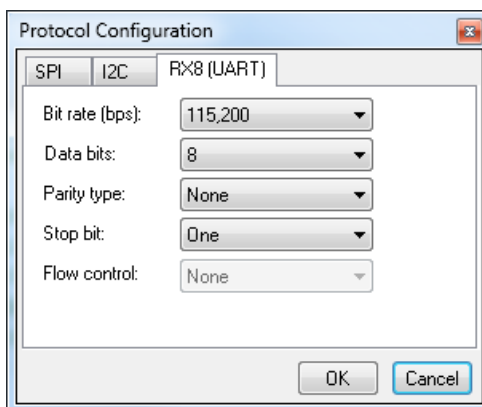
Figure 10. Main Window



2. Configure the UART protocol settings.

In the main window shown in Figure 10, select the required **COM** port under **Connected Ports**. RX8 (UART) is selected as protocol automatically. Now, select **Protocol Configuration** from the **Tools** menu. The settings dialog box appears as shown in Figure 11. Choose RX8 (UART) tab and configure the settings to match with the PSoC Designer project.

Figure 11. Settings Dialog Box



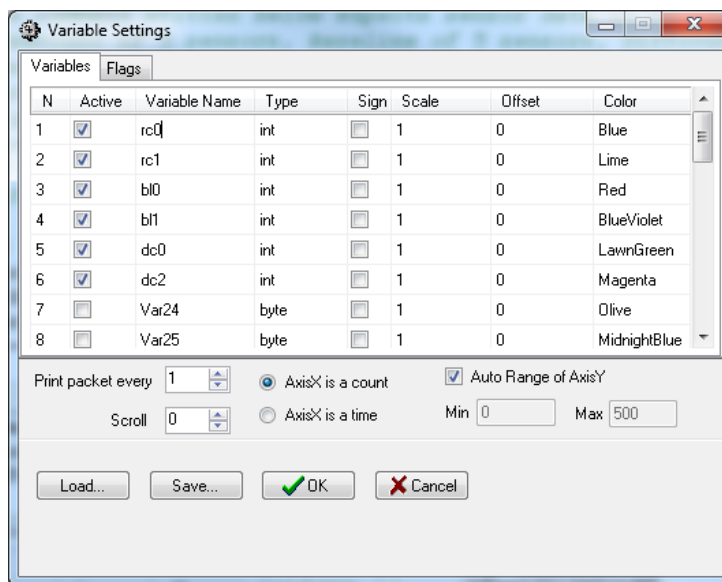
### 3. Monitor the CapSense data.

The following steps quickly show how to monitor the data using BCP. Refer to the BCP help manual (press F1) for more information.

- a. Add the variables in the Variable Settings window, which can be accessed at **Chart > Variable Settings** from the main window shown in Figure 10. To add a variable, check the **Active** box, edit the **Variable Name**, choose the **Type** (data type), and **Color**. Click **OK** when all the variables are added. The variable settings can be saved to a file by clicking **Save** and can be loaded later by clicking **Load**.

Figure 12 shows the variable settings window in which variables for Rawcount (rc0, rc1), Baseline (bl0, bl1), and Different count (dc0, dc1) added for two sensors.

Figure 12. Variable Settings Window



Enter the following RX8 command in the command area of the main window, as shown in Figure 10.

```
RX8 [H = 0D 0A] +
@1rc0 @0rc0 @1rc1 @0rc1 @1bl0 @0bl0 @1bl1 @0bl1 @1dc0 @0dc0 @1dc1 @0dc1 +
[T = 00 FF FF]
```

The command structure is explained below. ‘+’ symbol is used to split the command into multiple lines.

RX8	Command instructing BCP to expect UART data
[H = 0D 0A]	Header bytes
@1rc0...@0dc1	Data bytes
[T = 00 FF FF]	Tail bytes

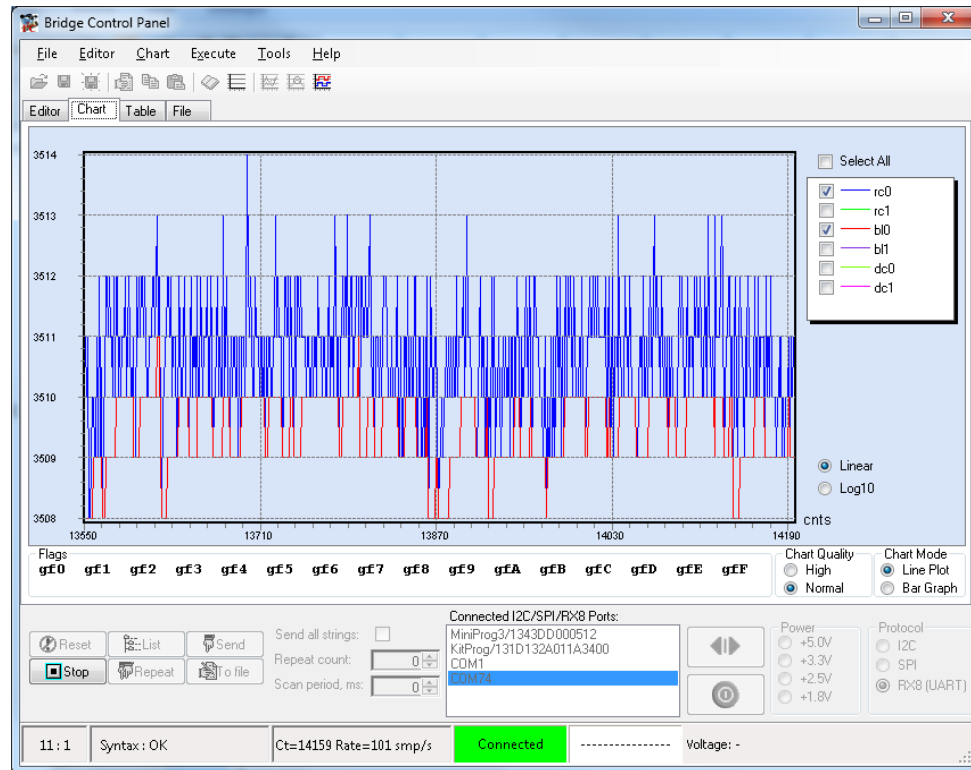
BCP looks at the data as byte stream. Therefore, a multibyte variable should be split into as many bytes as required and each byte should be represented as @Xvariablename, Where X is a number indicating the byte position in the variable.

For example, the MSB of the Rawcount (2 bytes) for sensor 0 is denoted as @1rc0 and the LSB as @0rc0. @1rc0 appears before @0rc0 in the command since MSB is sent out first.

- b. Select the entire RX8 command and click **Repeat**. The received data is displayed in the results window. Now, go to the **Chart** tab and select the required variables to monitor. Figure 13 shows plotting of Rawcount and Baseline for sensor 0.



Figure 13. Bridge Control Panel Chart Window



### 5.3 I<sup>2</sup>C with EZ-Click for CY8CMBR2110 and CY8CMBR3xxx Controllers

EZ-Click is a GUI-based tool that is used to configure the CY8CMBR2110 and CY8CMBR3xxx CapSense devices via an I<sup>2</sup>C interface.

#### 5.3.1 Step A: Install EZ-Click Tool

1. Download EZ-Click from [www.cypress.com/go/EZ-Click](http://www.cypress.com/go/EZ-Click) and install the software.
2. After installation, open the EZ-Click tool from the default location: **Start > All Programs > Cypress > EZ-Click 2.0 > EZ-Click 2.0.**

#### 5.3.2 Step B: Configure CY8CMBR2110/CY8CMBR3xxx Controller

Refer to the [EZ-Click 2.0 User Guide](#) for more information on these tasks:

1. Create a new project in the EZ-Click tool.
2. Specify various parameters for the selected CapSense controller and generate the configuration file.
3. Choose **Configuration > Select Target Device** to configure the CapSense controller, as shown in [Figure 14](#).

Depending on the kit ([CY3280-MBR2](#) kit or [CY3280-MBR3](#) kit) or the device you have connected to the PC, different **Ports** and **Devices** will be available to select, as shown in [Figure 15](#). Select the port to which the device is connected. Select an option for **Power** and **I<sup>2</sup>C Speed**, and click **OK**.

**Note:** The CY3280-MBR3 kit and CY3280-MBR2 kit come with a built-in I<sup>2</sup>C-to-USB bridge and can be directly connected to the PC using a USB A to Mini-B cable. If the CY8CMBR2110 or CY8CMBR3xxx device is placed on a custom board, you need to use either the [CY3240-I2USB](#) Bridge or the [MiniProg3](#) kit to connect the device to the PC.

Figure 14. Generating Configuration in EZ-Click

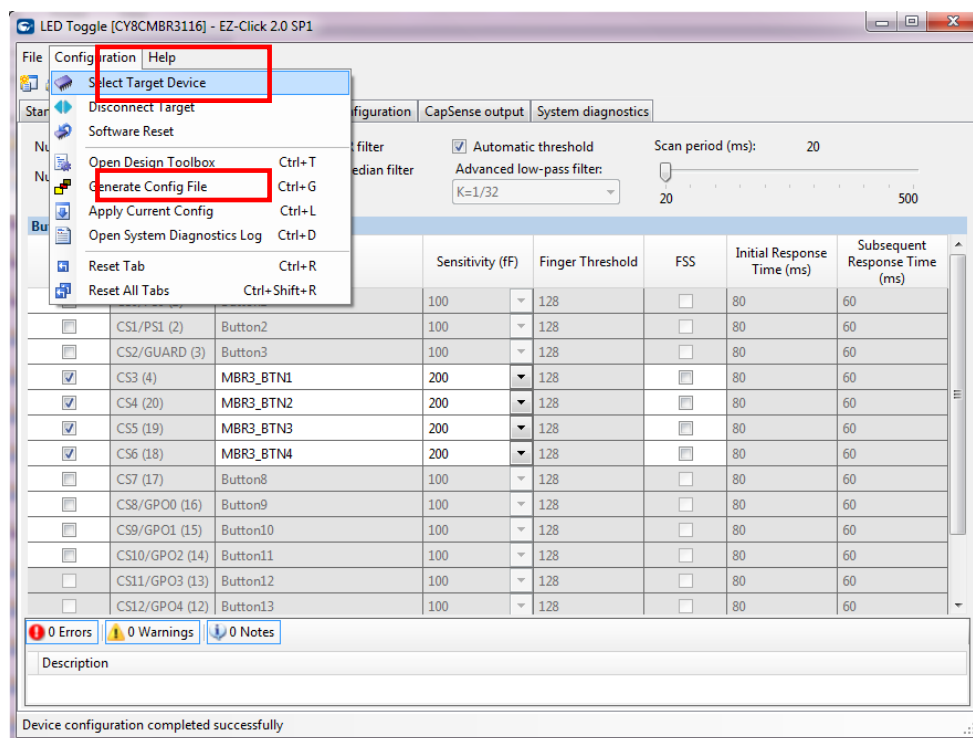
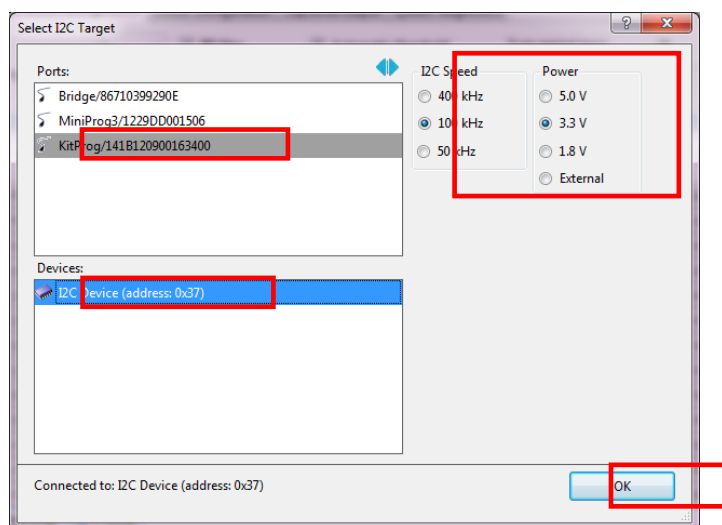
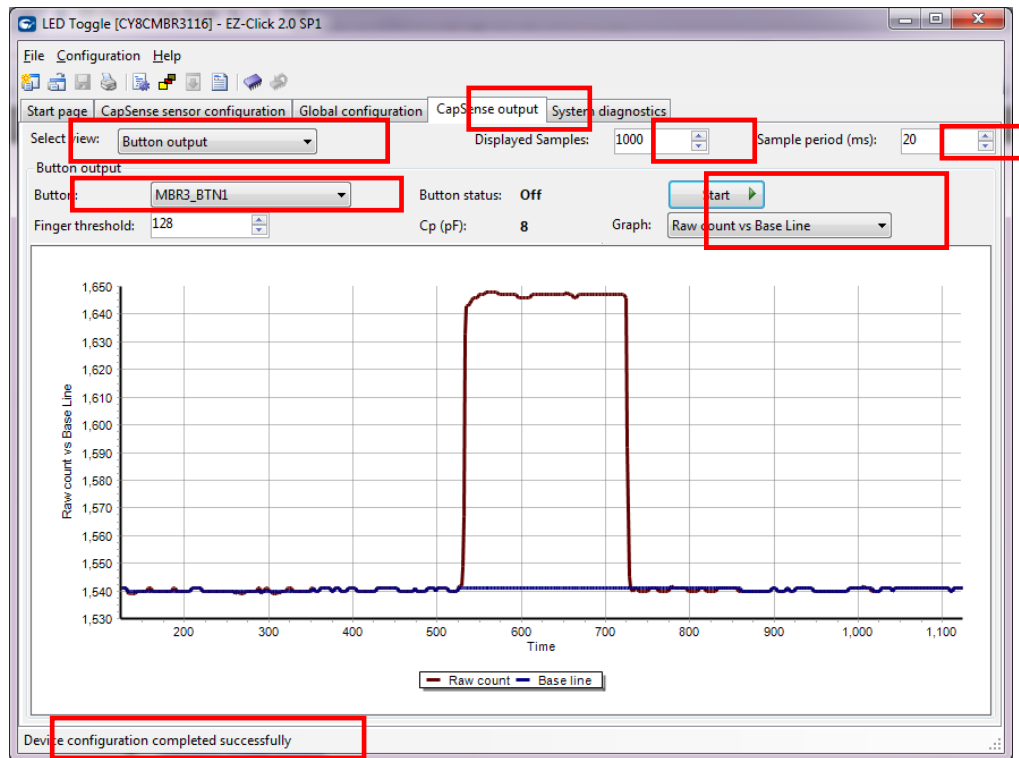


Figure 15. Selecting the Target Device in EZ-Click



4. Choose **Configuration > Generate Config File** to generate the configuration file.
5. Choose **Configuration > Apply Current Config** to configure the CapSense controller.
6. After the CapSense controller is successfully configured, click the **CapSense output** tab in the EZ-Click tool to view the CapSense sensor data, as shown in [Figure 16](#).

Figure 16. CapSense Output Tab in EZ-Click



### 5.3.3 Step C: View Raw Count and Baseline

1. To view the raw count and baseline, select the **Button Output** option in the **Select view** parameter, as shown in Figure 16.
2. In the **Button** parameter, specify the sensor for which the raw count and baseline data has to be displayed.
3. In the **Graph** parameter, select the **Raw count vs Baseline** option. Leave all other parameters at their default values.  
Refer to the [EZ-Click User Guide](#) for information on these parameters.
4. Click the **Start** button to acquire and display the raw count and baseline values of the sensor specified in the **Button** parameter.

To view the raw count value of all the sensors at the same time, select the **Parameter output** option in the **Select view** parameter and the **Raw count** option in the **Parameter** menu and click the **Start** button.

### 5.3.4 Step D: View Difference Count

The difference count value of a sensor is displayed in different windows depending on whether the Automatic threshold parameter (available in the **CapSense sensor configuration** tab) is enabled or disabled.

When the Automatic threshold parameter is enabled, follow this procedure to view the difference count:

1. In the **Select view** parameter, select the **Button Output** option.
2. In the **Graph** parameter, select the **Diff count vs Finger threshold** option and click the **Start** button to display the difference count of the sensor, as Figure 17 shows.

When the Automatic threshold parameter is disabled, follow this procedure to view the difference count:

1. In the **Select view** parameter, select the **Parameter output** option.

2. In the **Parameter** menu, select the **Difference count** option and click the **Start** button to display the difference count of the sensor, as Figure 18 shows.

Figure 17. Viewing Difference Count When Automatic Threshold Is Disabled

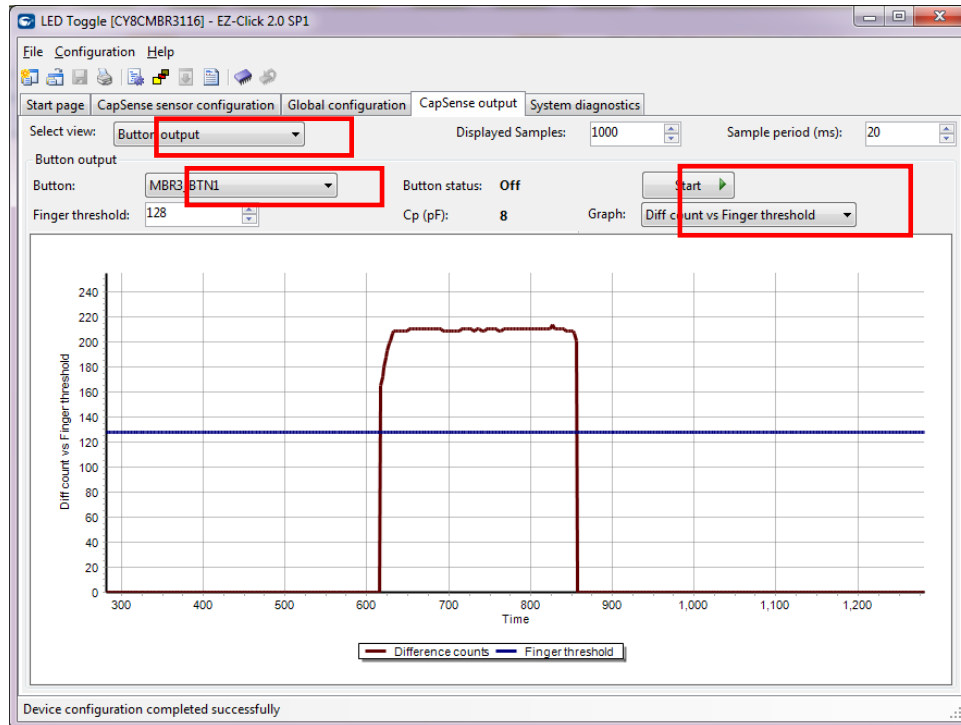
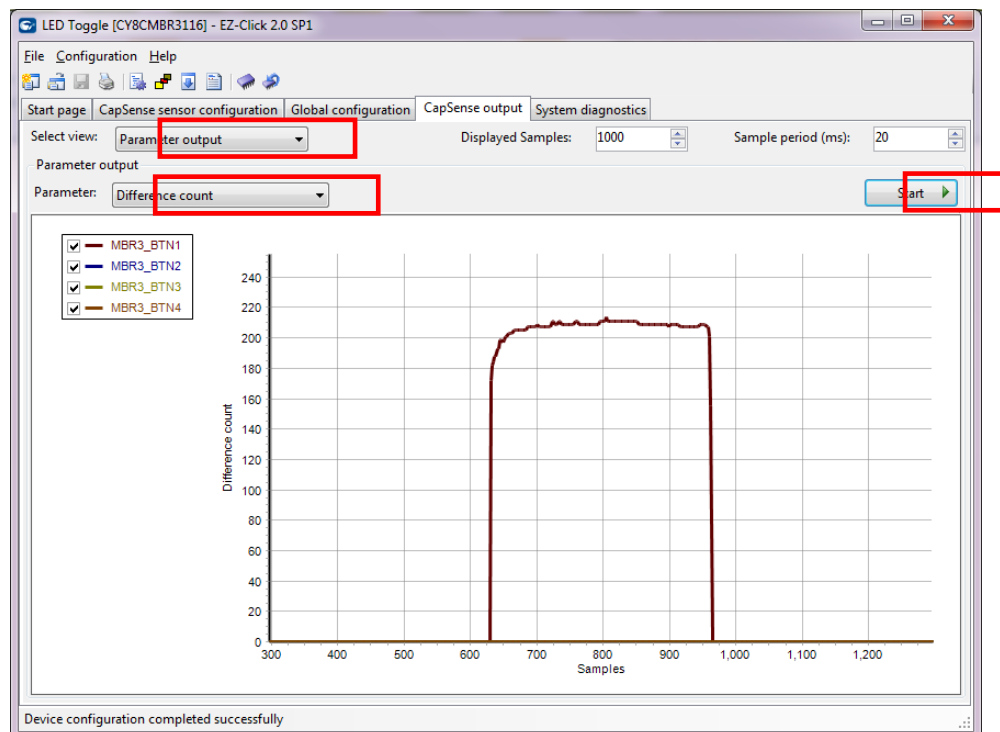


Figure 18. Viewing Difference Count When Automatic Threshold Is Enabled



## 5.4 UART with BCP for CY8CMBR20xx Controllers

### 5.4.1 System Requirements

This method requires the following system configuration:

- A free serial communication (COM/RS232) port <sup>[a]</sup>
- Microsoft Windows 9x, 2000, XP, or later

### 5.4.2 Supported Baud Rates

Unlike the programmable controllers, which can be programmed to send RS232 data at user-defined baud rates, the baud rates of CY8CMBR20xx devices are fixed. [Table 3](#) maps CY8CMBR20xx devices to the supported baud rate.

Table 3. Baud Rates Supported by CY8CMBR20xx

CY8CMBR20xx	Baud Rate
CY8CMBR2044	117.6 kbps
CY8CMBR2010/CY8CMBR2016	115.2 kbps

See the section [Enabling UART-to-USB Bridge Using CY3240-I2USB Bridge](#) for information about how to put together a UART-to-USB bridge to read RS232 data (clocked out at 117.6 kbps and 115.2 kbps).

Follow [Step A: Install BCP](#) and [Step D: Read CapSense Sensor Data Using BCP](#) to view CapSense data using this method.

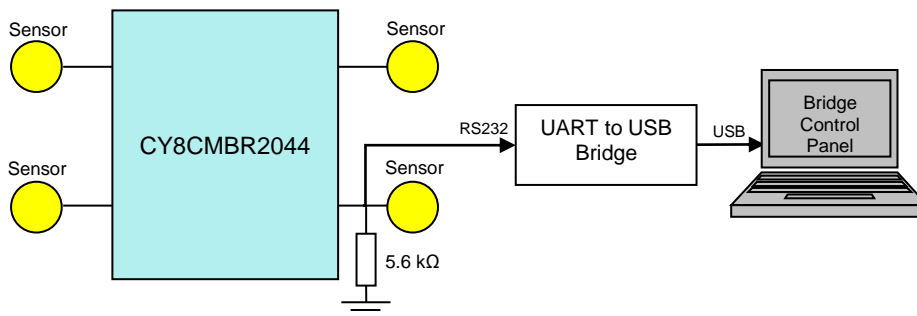
CY8CMBR20xx devices can be configured to send out the debug data on a pin. Refer to the corresponding device [datasheet](#) for the serial debug data format. Along with rawcount, baseline, and signal (difference count), CY8CMBR20xx devices also transmit the following information:

- Firmware revision
- CapSense button touch status
- GPO output status
- Parasitic capacitance of each CapSense sensor

### 5.4.3 Reading Data from CY8CMBR2044

CY8CMBR2044 transmits CapSense data in RS232 format (at a baud rate of 117.6 kbps) from the sensor pin, which is pulled LOW to GND with a 5.6-kΩ resistor. Refer to the device [datasheet](#) for the tolerance value of the resistor. [Figure 19](#) shows how to configure CY8CMBR2044 to send CapSense data.

Figure 19. CY8CMBR2044 Hardware Setup



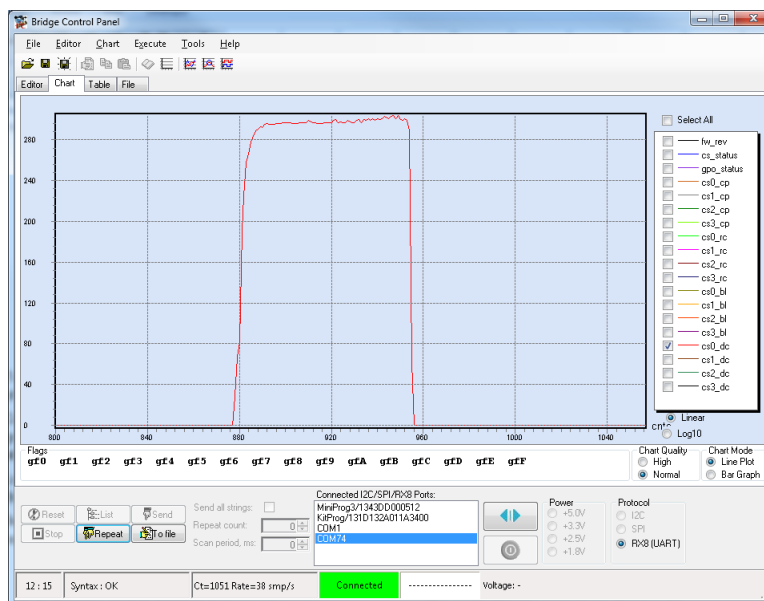
<sup>a</sup>This requirement is optional if the UART-to-USB bridge is used for reading serial (RS232) data.



1. To read data from a CY8CMBR2044 device, use a UART-to-USB bridge programmed to read RS232 data at a baud rate of 117.6 kbps. For details, see [Enabling UART-to-USB Bridge Using CY3240-I2USB Bridge](#).
2. Connect the CY8CMBR2044 sensor pin that is pulled low to GND to the I2C\_SDA pin of the CY3240-I2USB Bridge. Wire the GND terminal of CY8CMBR2044 to the GND pin of the CY3240-I2USB Bridge. See the [CY3240-I2USB Bridge](#) user guide for details about the CY3240-I2USB pinouts.
3. Refer to [Step D: Read CapSense Sensor Data Using BCP](#) to configure Bridge Control Panel for UART data monitoring.
4. Load the command file *CY8CMBR2044.iic* and the variable settings file *CY8CMBR2044.ini* provided with this application note into Bridge Control Panel.
5. Select the entire command and click the **Repeat** button to read the data. Now, go to the chart tab to monitor the required variable. [Figure 20](#) shows monitoring the different count of CS0.

Note that each bit in the sensor status and GPO status variables indicate the status of the corresponding sensor or the GPO. For example, the sensor status 0x08 indicates that a finger is detected on CS3; similarly the GPO status 0x04 indicates that GPO2 is asserted (active LOW).

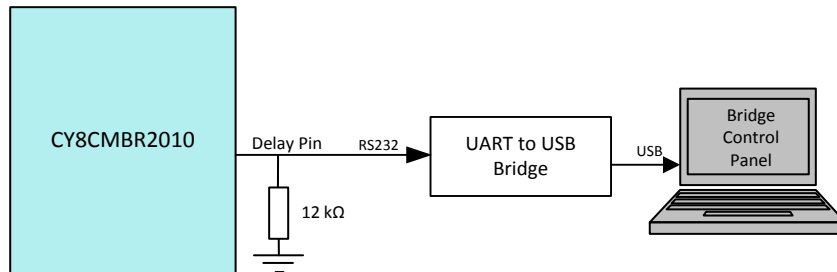
Figure 20. Monitoring Difference Count of CS0 for CY8CMBR2044



#### 5.4.4 Reading Data from CY8CMBR2010

CY8CMBR2010 transmits CapSense data in RS232 format (at a baud rate of 115.2 kbps) from the “Delay” pin, which is pulled LOW to GND with a 12-kΩ resistor. Refer to the device [datasheet](#) for the tolerance value of the resistor. [Figure 21](#) shows how to configure CY8CMBR2010 to send out CapSense data.

Figure 21. CY8CMBR2010 Hardware Setup

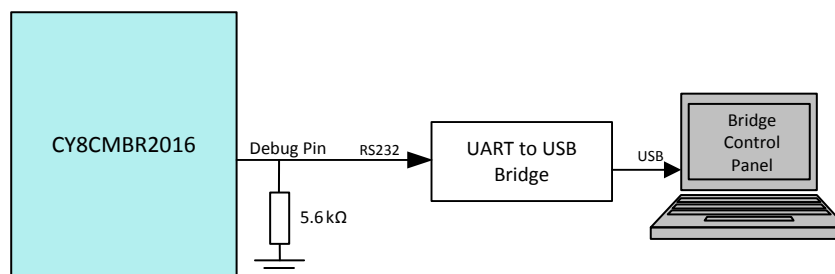


1. To read data from a CY8CMBR2010 device, use a UART-to-USB bridge programmed to read RS232 data at a baud rate of 115.2 kbps. For details, see [Enabling UART-to-USB Bridge Using CY3240-I2USB Bridge](#).
2. Connect the Delay pin of the CY8CMBR2010 to the I2C\_SDA pin of the CY3240-I2USB Bridge. Wire the GND terminal of CY8CMBR2010 to the GND pin of the CY3240-I2USB Bridge. See the [CY3240-I2USB Bridge](#) user guide for details about the CY3240-I2USB pinouts.
3. Refer to [Step D: Read CapSense Sensor Data Using BCP](#) to configure Bridge Control Panel for UART data monitoring.
4. Load the required command file *CY8CMBR2010\_xxxx.iic* and the variable settings file *CY8CMBR2010\_xxxx.ini* provided with this application note into Bridge Control Panel. All the data sent by CY8CMBR2010 cannot be monitored simultaneously using Bridge Control Panel because it can monitor only up to 32 named variables. However, any number of bytes can be read from the command window. Therefore, multiple command and variable setting files are provided for this device to selectively monitor the data. The following is the list of command files provided for this device. Each command file has a corresponding variable setting file.
  - *CY8CMBR2010\_SensorData.iic* – To monitor sensor status, GPO status, and CapSense scan data (Rawcount, Baseline, Difference count) of all 10 buttons.
  - *CY8CMBR2010\_DiagData.iic* – To monitor the remaining data: Firmware revision, Cp, SNR, and System diagnostics results
5. Select the entire command and click the **Repeat** button to read the data. Now, go to the chart tab to monitor the required variable.

#### 5.4.5 Reading Data from CY8CMBR2016

CY8CMBR2016 transmits CapSense data in RS232 format (at a baud rate of 115.2 kbps) from the “Debug” pin, which is pulled low to GND with a 5.6-kΩ resistor. Refer to the device [datasheet](#) for the tolerance value of the resistor. [Figure 22](#) shows how to configure CY8CMBR2016 to send CapSense data.

Figure 22. CY8CMBR2016 Hardware Setup



## Notes

1. To read data from a CY8CMBR2016 device, use a UART-to-USB bridge programmed to read RS232 data at a baud rate of 115.2 kbps. For details, see [Enabling UART-to-USB Bridge Using CY3240-I2USB Bridge](#).
2. Connect the Debug pin of the CY8CMBR2016 to the I2C\_SDA pin of the CY3240-I2USB Bridge. Wire the GND terminal of CY8CMBR2016 to the GND pin of the CY3240-I2USB Bridge. See the [CY3240-I2USB Bridge](#) user guide for details about the CY3240-I2USB pinouts.
3. Refer to [Step D: Read CapSense Sensor Data Using BCP](#) to configure Bridge Control Panel for UART data monitoring.
4. Load the required command file *CY8CMBR2016\_XXXX.iic* and the variable settings file *CY8CMBR2016\_XXXX.ini* provided with this application note into Bridge Control Panel. All the data sent by CY8CMBR2016 cannot be monitored simultaneously using Bridge Control Panel because it can monitor only up to 32 named variables. However, any number of bytes can be read from the command window. Therefore, multiple command and variable setting files are provided for this device to selectively monitor the data. The following is the list of command files provided for this device. Each command file has a corresponding variable setting file.
  - *CY8CMBR2016\_SensorData\_CS0toCS9.iic* – To monitor sensor status and CapSense scan data (Rawcount, Baseline, Difference count) for the sensors CS0 to CS9.
  - *CY8CMBR2016\_SensorData\_CS10toCS15.iic* – To monitor sensor status and CapSense scan data (Rawcount, Baseline, Difference count) for the sensors CS10 to CS15.
  - *CY8CMBR2016\_CpData.iic* – To monitor the firmware revision and Cp of all the sensors.
5. Select the entire command and click the **Repeat** button to read the data. Now go to the chart tab to monitor the required variable.

## 6 Enabling UART-to-USB Bridge Using CY3240-I2USB Bridge

This section covers the following two topics:

- How to convert CY3240-I2USB Bridge hardware to work as a UART-to-USB bridge to read RS232 data clocked out at 117.6 kbps and 115.2 kbps
- Step-by-step instructions for installing the UART-to-USB driver

### 6.1 Converting CY3240-I2USB to UART-to-USB Bridge

Program the CY3240-I2USB Bridge hardware with the *USB\_UART\_Bridge.hex* file.

- Get this file from the [BaudRate117.6\\_UART\\_TO\\_USB.zip](#) file for reading RS232 data clocked at 117.6 kbps.
- Get this file from the [BaudRate115.2\\_UART\\_TO\\_USB.zip](#) file for reading RS232 data clocked at 115.2 kbps.

For details on how to program the bridge, see the [CY3240-I2USB Bridge](#) user guide, section 3.2.1, “Program USB-I2C Bridge.”

### 6.2 UART-to-USB Driver Installation

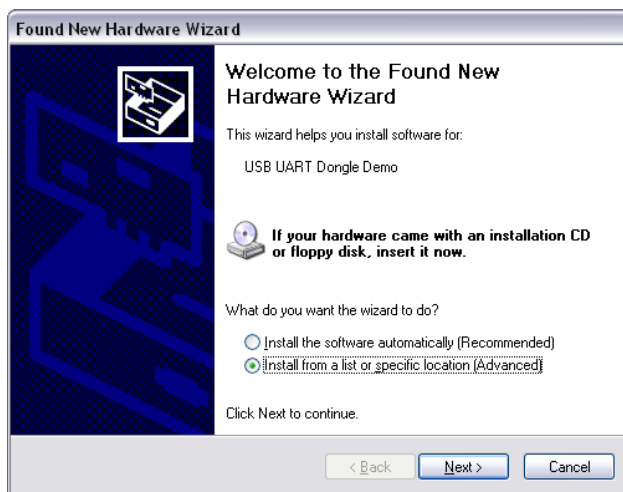
1. After you have finished programming, connect the CY3240-I2USB Bridge hardware to the PC USB port. When the device is first connected to the PC, the new hardware wizard starts.
2. Select **No, not this time** and click **Next**, as shown in [Figure 23](#).

Figure 23. New Hardware Wizard – Step 1



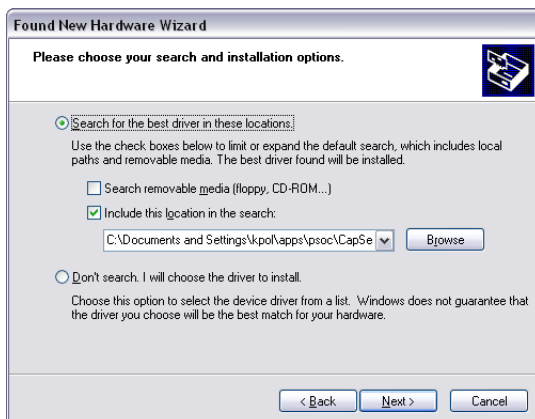
3. Select **Install from a list or specific location (Advanced)** and click **Next**, as shown in Figure 24.

Figure 24. New Hardware Wizard – Step 2



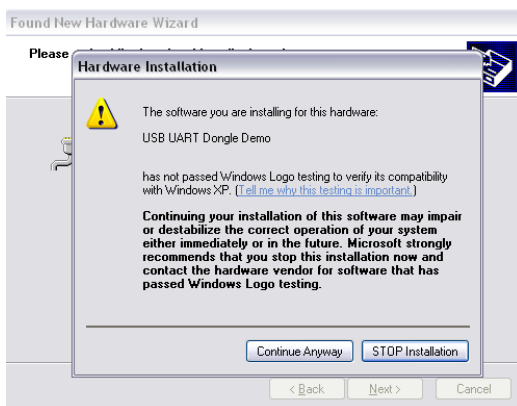
4. Select **Search for the best driver in these locations**, and **Include this location in the search** and set the path to point to the folder containing the `UART_TO_USB.hex` file. Then click **Next**, as shown in Figure 25.

Figure 25. New Hardware Wizard – Step 3



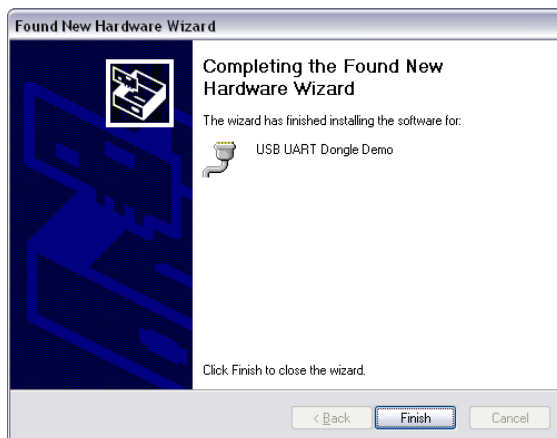
5. Click **Continue Anyway**, as shown in Figure 26.

Figure 26. New Hardware Wizard – Step 4



6. Click **Finish** to complete the driver installation, as shown in Figure 27.

Figure 27. New Hardware Wizard – Step 5





## 7 Glossary

### **AMUXBUS**

Analog multiplexer bus available inside PSoC that helps to connect I/O pins with multiple internal analog signals.

### **SmartSense™ Auto-Tuning**

A CapSense algorithm that automatically sets sensing parameters for optimal performance after the design phase and continuously compensates for system, manufacturing, and environmental changes.

### **Baseline**

A value resulting from a firmware algorithm that estimates a trend in the Raw Count when there is no human finger present on the sensor. The Baseline is less sensitive to sudden changes in the Raw Count and provides a reference point for computing the Difference Count.

### **Button or Button Widget**

A widget with an associated sensor that can report the active or inactive state (that is, only two states) of the sensor. For example, it can detect the touch or no-touch state of a finger on the sensor.

### **Difference Count**

The difference between Raw Count and Baseline. If the difference is negative, or if it is below Noise Threshold, the Difference Count is always set to zero.

### **Capacitive Sensor**

A conductor and substrate, such as a copper button on a printed circuit board (PCB), which reacts to a touch or an approaching object with a change in capacitance.

### **CapSense®**

Cypress's touch-sensing user interface solution. The industry's No. 1 solution in sales by 4x over No. 2.

### **CapSense Mechanical Button Replacement (MBR)**

Cypress's configurable solution to upgrade mechanical buttons to capacitive buttons, requires minimal engineering effort to configure the sensor parameters and does not require firmware development. These devices include the CY8CMBR3XXX and CY8CMBR2XXX families.

### **Centroid or Centroid Position**

A number indicating the finger position on a slider within the range given by the Slider Resolution. This number is calculated by the CapSense centroid calculation algorithm.

### **Compensation IDAC**

A programmable constant current source, which is used by CSD to compensate for excess sensor  $C_p$ . This IDAC is not controlled by the Sigma-Delta Modulator in the CSD block unlike the Modulation IDAC.

### **CSD**

CapSense Sigma Delta (CSD) is a Cypress-patented method of performing self-capacitance (also called self-cap) measurements for capacitive sensing applications.

In CSD mode, the sensing system measures the self-capacitance of an electrode, and a change in the self-capacitance is detected to identify the presence or absence of a finger.

**Debounce**

A parameter that defines the number of consecutive scan samples for which the touch should be present for it to become valid. This parameter helps to reject spurious touch signals.

A finger touch is reported only if the Difference Count is greater than Finger Threshold + Hysteresis for a consecutive Debounce number of scan samples.

**Driven-Shield**

A technique used by CSD for enabling liquid tolerance in which the Shield Electrode is driven by a signal that is equal to the sensor switching signal in phase and amplitude.

**Electrode**

A conductive material such as a pad or a layer on PCB, ITO, or FPCB. The electrode is connected to a port pin on a CapSense device and is used as a CapSense sensor or to drive specific signals associated with CapSense functionality.

**Finger Threshold**

A parameter used with Hysteresis to determine the state of the sensor. Sensor state is reported ON if the Difference Count is higher than Finger Threshold + Hysteresis, and it is reported OFF if the Difference Count is below Finger Threshold – Hysteresis.

**Ganged Sensors**

The method of connecting multiple sensors together and scanning them as a single sensor. Used for increasing the sensor area for proximity sensing and to reduce power consumption.

To reduce power when the system is in low-power mode, all the sensors can be ganged together and scanned as a single sensor taking less time instead of scanning all the sensors individually. When the user touches any of the sensors, the system can transition into active mode where it scans all the sensors individually to detect which sensor is activated.

PSoC supports sensor-ganging in firmware, that is, multiple sensors can be connected simultaneously to AMUXBUS for scanning.

**Gesture**

Gesture is an action, such as swiping and pinch-zoom, performed by the user. CapSense has a gesture detection feature that identifies the different gestures based on predefined touch patterns. In the CapSense component, the Gesture feature is supported only by the Touchpad Widget.

**Guard Sensor**

Copper trace that surrounds all the sensors on the PCB, similar to a button sensor and is used to detect a liquid stream. When the Guard Sensor is triggered, firmware can disable scanning of all other sensors to prevent false touches.

**Hatch Fill or Hatch Ground or Hatched Ground**

While designing a PCB for capacitive sensing, a grounded copper plane should be placed surrounding the sensors for good noise immunity. But a solid ground increases the parasitic capacitance of the sensor which is not desired. Therefore, the ground should be filled in a special hatch pattern. A hatch pattern has closely-placed, crisscrossed lines looking like a mesh and the line width and the spacing between two lines determine the fill percentage. In case of liquid tolerance, this hatch fill referred as a shield electrode is driven with a shield signal instead of ground.

**Hysteresis**

A parameter used to prevent the sensor status output from random toggling due to system noise, used in conjunction with the Finger Threshold to determine the sensor state. See [Finger Threshold](#).

**IDAC (Current-Output Digital-to-Analog Converter)**

Programmable constant current source available inside PSoC, used for CapSense and ADC operations.

**Liquid Tolerance**

The ability of a capacitive sensing system to work reliably in the presence of liquid droplets, streaming liquids or mist.

**Linear Slider**

A widget consisting of more than one sensor arranged in a specific linear fashion to detect the physical position (in single axis) of a finger.

**Low Baseline Reset**

A parameter that represents the maximum number of scan samples where the Raw Count is abnormally below the Negative Noise Threshold. If the Low Baseline Reset value is exceeded, the Baseline is reset to the current Raw Count.

**Manual-Tuning**

The manual process of setting (or tuning) the CapSense parameters.

**Matrix Buttons**

A widget consisting of more than two sensors arranged in a matrix fashion, used to detect the presence or absence of a human finger (a touch) on the intersections of vertically and horizontally arranged sensors.

If M is the number of sensors on the horizontal axis and N is the number of sensors on the vertical axis, the Matrix Buttons Widget can monitor a total of M x N intersections using ONLY M + N port pins.

When using the CSD sensing method (self-capacitance), this Widget can detect a valid touch on only one intersection position at a time.

**Modulation Capacitor (CMOD)**

An external capacitor required for the operation of a CSD block in Self-Capacitance sensing mode.

**Modulator Clock**

A clock source that is used to sample the modulator output from a CSD block during a sensor scan. This clock is also fed to the Raw Count counter. The scan time (excluding pre and post processing times) is given by  $(2^N - 1) / \text{Modulator Clock Frequency}$ , where N is the Scan Resolution.

**Modulation IDAC**

Modulation IDAC is a programmable constant current source, whose output is controlled (ON/OFF) by the sigma-delta modulator output in a CSD block to maintain the AMUXBUS voltage at  $V_{REF}$ . The average current supplied by this IDAC is equal to the average current drawn out by the sensor capacitor.

**Mutual-Capacitance**

Capacitance associated with an electrode (say TX) with respect to another electrode (say RX) is known as mutual capacitance.

**Negative Noise Threshold**

A threshold used to differentiate usual noise from the spurious signals appearing in negative direction. This parameter is used in conjunction with the Low Baseline Reset parameter.

Baseline is updated to track the change in the Raw Count as long as the Raw Count stays within Negative Noise Threshold, that is, the difference between Baseline and Raw count (Baseline – Raw count) is less than Negative Noise Threshold.

Scenarios that may trigger such spurious signals in a negative direction include: a finger on the sensor on power-up, removal of a metal object placed near the sensor, removing a liquid-tolerant CapSense-enabled product from the water; and other sudden environmental changes.

**Noise (CapSense Noise)**

The variation in the Raw Count when a sensor is in the OFF state (no touch), measured as peak-to-peak counts.

**Noise Threshold**

A parameter used to differentiate signal from noise for a sensor. If Raw Count – Baseline is greater than Noise Threshold, it indicates a likely valid signal. If the difference is less than Noise Threshold, Raw Count contains nothing but noise.

**Overlay**

A non-conductive material, such as plastic and glass, which covers the capacitive sensors and acts as a touch-surface. The PCB with the sensors is directly placed under the overlay or is connected through springs. The casing for a product often becomes the overlay.

**Parasitic Capacitance ( $C_P$ )**

Parasitic capacitance is the intrinsic capacitance of the sensor electrode contributed by PCB trace, sensor pad, vias, and air gap. It is unwanted because it reduces the sensitivity of CSD.

**Proximity Sensor**

A sensor that can detect the presence of nearby objects without any physical contact.

**Radial Slider**

A widget consisting of more than one sensor arranged in a specific circular fashion to detect the physical position of a finger.

**Raw Count**

The unprocessed digital count output of the CapSense hardware block that represents the physical capacitance of the sensor.

**Refresh Interval**

The time between two consecutive scans of a sensor.

**Scan Resolution**

Resolution (in bits) of the Raw Count produced by the CSD block.

**Scan Time**

Time taken for completing the scan of a sensor.

**Self-Capacitance**

The capacitance associated with an electrode with respect to circuit ground.

**Sensitivity**

The change in Raw Count corresponding to the change in sensor capacitance, expressed in counts/pF. Sensitivity of a sensor is dependent on the board layout, overlay properties, sensing method, and tuning parameters.

**Sense Clock**

A clock source used to implement a switched-capacitor front-end for the CSD sensing method.

**Sensor**

See [Capacitive Sensor](#).

**Sensor Auto Reset**

A setting to prevent a sensor from reporting false touch status indefinitely due to system failure, or when a metal object is continuously present near the sensor.

When Sensor Auto Reset is enabled, the Baseline is always updated even if the Difference Count is greater than the Noise Threshold. This prevents the sensor from reporting the ON status for an indefinite period of time. When Sensor Auto Reset is disabled, the Baseline is updated only when the Difference Count is less than the Noise Threshold.

**Sensor Ganging**

See [Ganged Sensors](#).

**Shield Electrode**

Copper fill around sensors to prevent false touches due to the presence of water or other liquids. Shield Electrode is driven by the shield signal output from the CSD block. See [Driven-Shield](#).

**Shield Tank Capacitor (C<sub>SH</sub>)**

An optional external capacitor (C<sub>SH</sub> Tank Capacitor) used to enhance the drive capability of the CSD shield, when there is a large shield layer with high parasitic capacitance.

**Signal (CapSense Signal)**

Difference Count is also called Signal. See Difference Count.

**Signal-to-Noise Ratio (SNR)**

The ratio of the sensor signal, when touched, to the noise signal of an untouched sensor.

**Slider Resolution**

A parameter indicating the total number of finger positions to be resolved on a slider.

**Touchpad**

A Widget consisting of multiple sensors arranged in a specific horizontal and vertical fashion to detect the X and Y position of a touch.

**Trackpad**

See [Touchpad](#).

**Tuning**

The process of finding the optimum values for various hardware and software or threshold parameters required for CapSense operation.

**V<sub>REF</sub>**

Programmable reference voltage block available inside PSoC used for CapSense and ADC operation.

**Widget**

A user-interface element in the CapSense component that consists of one sensor or a group of similar sensors. Button, proximity sensor, linear slider, radial slider, matrix buttons, and touchpad are the supported widgets.

## 8 Summary

This application note shows how to monitor CapSense debug data from Cypress CapSense controllers using the I<sup>2</sup>C or UART interface. Detailed explanation is provided on how to monitor the CapSense data using Bridge Control Panel and EZ-Click tools.

---

## About the Author

Name: Kurian Polachan

Title: Sr. Applications Engineer

## Document History

Document Title: AN2397 - PSoC® 1 and CapSense® Controllers – CapSense Data Monitoring Tools

Document Number: 001-41446

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1541809	KPOL	10/09/2007	New application note
*A	3124643	ARVM	01/03/2011	Included CY8C20xx6A in the "Associated Part Family". Fixed the broken links. Deleted references to CSR UM
*B	3404186	KPOL	10/14/2011	Complete rewrite
*C	3461633	KPOL	12/21/2011	Updated template to current Cypress standards. Fixed typos.
*D	3804024	VAIR	11/06/2012	Added CY8C20xx7/S. Updated links to code examples. Added link to PSoC 3 and PSoC 5 Tuner GUI.
*E	3902185	VAIR	02/12/2013	Changed description of Figure 17.
*F	4096713	VAIR	08/15/2013	Added reference and link to datasheet.
*G	4478381	DCHE	08/20/2014	Added section <a href="#">I2C with EZ-Click</a>
*H	4581694	DCHE	11/27/2014	Renamed CY3240-I2USB Kit as CY3240-I2USB Bridge. Updated hyperlinks for CY3240-I2USB Bridge Guide.
*I	4787203	VAIR	06/24/2015	Major rewrite and Multichart is replaced with Bridge Control Panel for UART based monitoring Added section <a href="#">CapSense Resources</a> Updated to new template.
*J	5094088	VAIR	01/20/2016	Added Glossary.

## 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

Automotive	<a href="http://cypress.com/go/automotive">cypress.com/go/automotive</a>
Clocks & Buffers	<a href="http://cypress.com/go/clocks">cypress.com/go/clocks</a>
Interface	<a href="http://cypress.com/go/interface">cypress.com/go/interface</a>
Lighting & Power Control	<a href="http://cypress.com/go/powerpsoc">cypress.com/go/powerpsoc</a>
Memory	<a href="http://cypress.com/go/memory">cypress.com/go/memory</a>
PSoC	<a href="http://cypress.com/go/psoc">cypress.com/go/psoc</a>
Touch Sensing	<a href="http://cypress.com/go/touch">cypress.com/go/touch</a>
USB Controllers	<a href="http://cypress.com/go/usb">cypress.com/go/usb</a>
Wireless/Rf	<a href="http://cypress.com/go/wireless">cypress.com/go/wireless</a>

## PSoC® Solutions

[psoc.cypress.com/solutions](http://psoc.cypress.com/solutions)

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

## Cypress Developer Community

[Community](#) | [Forums](#) | [Blogs](#) | [Video](#) | [Training](#)

## Technical Support

[cypress.com/go/support](http://cypress.com/go/support)

PSoC is a registered trademark and PSoC Creator is a trademark of Cypress Semiconductor Corp. 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

Phone : 408-943-2600  
Fax : 408-943-4730  
Website : [www.cypress.com](http://www.cypress.com)

© Cypress Semiconductor Corporation, 2007-2016. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.